1985-09

SWOPATH: an interactive network flow model simulating the U.S. Navy Surface Warfare Officer Career Paths

Amirault, Richard Bradford

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SWOPATH: AN INTERACTIVE NETWORK
FLOW MODEL SIMULATING THE
U.S. NAVY SURFACE WARFARE OFFICER
CAREER PATHS

by

Richard Bradford Amirault

September 1985

Thesis Advisor: P. R. Milch

Approved for public release; distribution is unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): SWOPATH: An Interactive Network Flow Model Simulating the U.S.Navy Surface Warfare Officer Career Paths	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, September 1985
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Richard Bradford Amirault	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	12. REPORT DATE: September 1985
	13. NUMBER OF PAGES: 137
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Network Flow Model
Manpower Modelling
Interactive Computer Model
Career Paths
Career Development
Surface Warfare Officer
Career Planning

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents an interactive computer model designed to examine the Surface Warfare Officer (SWO) Career Path. The model called SWOPATH is designed to provide the manpower managers with a fast, user friendly analytical tool. The model is derived from a network representation of a SWO career path, the rows representing the billet activities, the columns the tours of duty. Career paths are represented

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

by arcs connecting the nodes of the network. The model allows the manager to display current data and evaluate the effect of altering career paths by changing assignment tour lengths, transfer percentages from assignments to assignments, and accessions to the SWO community. The model's speed allows the manager to evaluate several scenarios, providing an ability to quickly forecast results of policy changes on the SWO community.

Approved for public release; distribution is unlimited.

SWOPATH: An Interactive Network Flow Model Simulating the U.S. Navy Surface Warfare Officer Career Paths

by

Richard Bradford Amirault
Lieutenant Commander, United States Navy
B.S., Boston College, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

This thesis presents an interactive computer model designed to examine the Surface Warfare Officer (SWO) Career Path. The model called SWOPATH is designed to provide the manpower managers with a fast, user friendly analytical tool. The model is derived from a network representation of a SWO career path, the rows representing the billet activities, the columns the tours of duty. Career paths are represented by arcs connecting the nodes of the network. The model allows the manager to display current data and evaluate the effect of altering career paths by changing assignment tour lengths, transfer percentages from assignments to assignments, and accessions to the SWO community. The model's speed allows the manager to evaluate several scenarios, providing an ability to quickly forecast results of policy changes on the SWO community.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

The Unrestricted Line Officer Career Planning Guidebook describes the Surface Warfare Community as follows:

> The Surface Warfare Community is composed of officers who are qualified in the surface warfare speciality, who man the surface ships of the Navy and whose goal is to command those ships. The Surface Warfare Officer (SWO) must, through a progression of competitive assignments, learn the fundamentals of engineering, weapons systems, and operational tactics. [Ref. 1: p. 23]

In order to obtain these qualifications, and attain the experience and knowledge necessary to reach the goal of command at sea, each surface warfare officer proceeds along a sequence of various assignments over a career. This sequence of assignments is usually described as a career path. There are many factors involved in detailing officers to their assignments, resulting in many different career paths. However it is possible to classify career paths into a small number of groups within which all paths exhibit the same basic features. In an NPS thesis entitled "The Effect of PCS Policy Changes on Surface Warfare Officer (SWO) Career Development" by R. H. HOWE, [Ref. 2], a network representation of SWO career paths is developed and used for categorizing them. The network is arranged in a rectangular array of nodes, where the rows stand for the general activities to which officers are assigned, and the columns represent the tour sequences, starting with the first, through the twelfth tour, in the career of a surface warfare officer. Each node in the network represents an assignment of a surface warfare officer, specified by an activity and a tour number.

B. PURPOSE

The purpose of this thesis is to use this "network representation" of the surface warfare officer's career path in developing a computer supported tool which would simulate, or model the movement or flow of surface warfare officers within this network. This decision support tool would assist the analyst in evaluating the impact of changes in policy that affect the surface warfare officer's career path. By using this model an analyst can address "what if" type questions about tour length changes or changes to the career paths leading up to major command. For instance the model might be used to answer the question: "If we increased the tour at department head school to one year vice the current six months, what impact would that have in maintaining a sufficient number of officers available to fill executive officer (XO) billets in future years?" To answer this question now, the analyst would need to figure out how many XO billets were needed to be filled, then calculate the effect of changing the number of officers leaving department head school to go on to becoming an XO via several in between tours. The analyst would need to determine how many officers were transferred to where, and perform several hand calculations to determine whether all the XO billets would be filled. With the speed of a computer model the program allows the analyst to perform the same calculations much faster and with greater assurance of accuracy. The time saved could now be used to evaluate many more scenarios than time would permit previously.

C. THE MODEL

The model is designed to provide a simulation of the flows through the network, representing the surface warfare officers career paths, dependent on user input. The model is interactive with the user, and provides a menu of choices for the user to select from, when proceeding through the

model. The model is written as a series of procedures, each of which are called upon by the user to carry out various functions within the model. These procedures are invisible to the user while he is operating the model. Complete listings of the procedures are provided in Appendix A through N.

The model is written with both a new and an experienced user in mind. Wherever there is an opportunity for the user to make a mistake, there is an error trapping procedure that simply provides the user with a displayed warning, indicating that an invalid response was made, and the user is returned to the exact spot in the program where he made the error.

The user is also provided the opportunity to choose whether to use default data provided with the model to initialize the variables in the network, or to use files that the user may have created when previously operating the model.

The model provides for a display of data in several formats. For example, the number of officers at a particular tour for all activities allows the user to review the data and evaluate the results of the simulation run to be explained below. By utilizing the "PrintScreen" function (if available) on the user's micro computer, the user can make hard copies of the displays for future reference.

The model simulates the flow of officers through the network for a time period chosen by the user. The model advances officers in time units of quarters, i.e. it moves officers from assignments to assignments each quarter. Selecting to run the simulation, for example, for two years means that the model will simulate flows for eight quarters.

If the analyst were to use the model to answer the question posed above, he would change the tour length of the assignments that related to department head school, and then

establish a limit for the number of XO billets which if violated would cause a warning message to be displayed. For example, if the analyst is interested in having at least 400 XO's, then the analyst would want to know when there were less than 400 XO's available. The analyst chooses the number of years to run the model, and observes the results. If in fact the change to the tour length of department head school reduces the number of officers available for XO billets at some future time, then when the number of XO's drops below 400 the model will display a warning message that will indicate how long the simulation ran before the parameter was violated and how many officers currently are present. The analyst can then decide whether to make such a change in the tour length. If the change is still desired, the analyst will need to evaluate the effect of other changes to the normal career path, e.g. decrease some other tour length after department head school, or increase the number of officers being transferred to XO billets, or some other combination of changes, to insure that once the tour length of department head school has been altered, the XO billets will still be filled. The analyst would be able to utilize the data display capabilities of the model to evaluate his alternatives.

This model was designed with the increased availability of micro computers in mind. Written and compiled using Turbo Pascal Version 3.0 (Trademark of Borland International), this model comes in two versions. One version is written to be used on any IBM PC compatible micro computer, and another version for the Heath/Zenith Models 100/110/120 micro computers. Both versions of the model are written for color monitors but will run on black and white monitors without any changes by the user. Because of the increasing availability of micro computers both in the workplace and in the home, the portability of this model should provide for increased usage, without the inherent

problems of waiting for terminal availability.

In summary, the model with data chosen by the user, simulates the flow of officers through a typical career path for any number of years ahead. The results can be displayed at the user's options. Data/parameters can be changed by the user and another simulation can be run at the user's option. Also at the user's option, data created during the simulation, or data/parameters that have been changed by the user, can be saved to files named by the user, for future use. The model is highly user friendly, with short warnings when invalid responses have been chosen. All data can be displayed in usable formats. The model thus provides the user with virtually an unlimited number of options to evaluate policy changes affecting the surface warfare officer's career path.

II. ANALYTICAL DESCRIPTION OF THE MODEL

The model described in this thesis consists of a network of nodes arranged in rows and columns, using either the default data provided with the model, or data created by the user as he operates the model. The network's rows represent eight activities, such as sea duty or duty in Washington DC, to which a surface warfare officer might be ordered. Each of the network's columns represent one of twelve possible tours of duty. As shown in Figure 2.1, this creates an eight by twelve network of nodes. Each node in the network represents an assignment where a surface warfare officer might be sent, specified by an activity and a tour number at which that assignment occurs. For example, the node "3A" in Figure 2.1, would denote an assignment occurring during the third tour of duty, at activity "A". Each node also has a number attached representing the number of officers so assigned. Each node or assignment has a specific tour length in quarters. From each assignment, there are arcs or transfer paths to any one of the assignments in the next column. For example, node "3A" would have eight possible transfer paths, to nodes "4A" through "4H". To each transfer path there is a percentage attached which represents the percentage of officers transferred from the source node at an end of the arc to the destination node at the other end of the arc. Each assignment also has a high and a low limit constraining the number of officers so assigned. For example, the node "3A" might have a high limit of 150 officers, and a low limit of 50 officers. The program will indicate to the user when a limit is violated.

PRO TRNG	1A	2A	3A	4A	5A	6A	7A	8A	9A	10A	11A	12A
PRO ED	1B	2B	3B	4B	5B	6B	7B	8B	9B	10B	11B	12B
WASH DC	1C	2C	3C	4C	5C	6C	7C	8C	9C	10C	11C	12C
SHORE CONUS	1D	2D	3D	4D	5D	6D	7D	8D	9D	10D	11D	12D
FLEET UNIT	1E	2E	3E	4E	5E	6E	7E	8E	9E	10E	11E	12E
AFLOAT STAFF	1F	2F	3F	4F	5F	6F	7F	8F	9F	10F	11F	12F
SHORE OUTUS	1G	2G	3G	4G	5G	6G	7G	8G	9G	10G	11G	12G
SEPAR-ATION	1H	2H	3H	4H	5H	6H	7H	8H	9H	10H	11H	12H

Figure 2.1 Network Representation

A. THE MODEL'S PARTS

1. Activities

In this network, the rows are representing generic activities to which a surface warfare officer may be assigned during his career. These activities are categorized as follows:

a. Professional Training

Student billets in either the SWO Department Head or SWO (Basic) courses longer than 20 weeks.

b. Professional Education

Student billets at a postgraduate school or a war or staff college of duration longer than 20 weeks.

c. Washington DC Tour

Shore duty billets in the Washington metropolitan area not meeting any of the criteria in (a) or (b) above.

d. Shore (CONUS)

Shore duty billets within the continental United States not meeting any of the criteria in (a), (b), or (c) above.

e. Fleet Unit

Ship's company sea duty billets.

f. Afloat Staff

Afloat staff sea duty billets.

g. Shore (OUTUS)

Non-CONUS shore duty billets.

h. Separation

Loss of officers to the SWO community. The main reasons are resignation (voluntary and involuntary), retirement, and lateral transfer to another officer community. This last activity, SEPARATION, is included in order to provide an "assignment" to account for attrition of officers.

2. Tours

Each duty station along the SWO career path that is longer than 20 weeks in duration is called a tour of duty. The model identifies these as tours starting with the first tour and progressing through the twelfth tour. The tours are represented by the columns in the network.

3. Billet Nodes/Assignments

Officers are detailed or ordered to specific assignments. These assignments can be described by specifying an activity and the number of the tour. For example, department head school during an officer's third tour of duty, would be defined in the model as "Third Tour Professional Training." Assignments are represented by the nodes in the network. Accessions to the surface warfare community are accounted for by the number of officers assigned to a specific node, namely the node "First Tour Fleet" which is where all officers enter the surface warfare community from all sources, the U.S. Naval Academy, Officer Candidate School (OCS), Navy Reserve Officer Training Courses (NROTC), etc.

4. Tour Lengths

Each assignment, depending on the activity and when the assignment occurs during the officer's career path, is for a specific length of time. The model defines these as "Tour Lengths" and allows for each possible assignment in the network to be up to 16 quarters (48 months) long, in units of quarters. For example, department head school during an officer's third tour is two quarters (6 months) long.

5. Arcs/Transfer Paths

From each assignment, with the exception of the activity, "SEPARATION", the officer proceeds to one of eight possible assignments in the next column of the network. For example, from the "Third Tour Professional Training", the officer may proceed to another professional training, or a

professional education, or a tour in Washington, or shore duty (CONUS), or a fleet unit, or an afloat staff, or shore duty (OUTUS), or possibly separation. The number of officers proceeding to each of the next assignments may be expressed as a percentage of the total number of officers leaving their last assignments. These percentages will be called "Transfer Path Percentages" in the model, and range from 0 to 100, depending on how many officers can be expected to be ordered from one particular assignment to a specific next assignment during each quarter. Each billet node or assignment in the network (except those in the last row), has eight possible transfer paths associated with it. Some transfer path percentages assigned to some arcs are 0, meaning that no one is transferred along that transfer path, and some may be 100, meaning that all officers are transferred along that path.

6. High and Low Limits

Certain assignments are limited in the number of officers that can be assigned to them at any one time. For example, department head school may be physically limited to approximately 150 seats. The model provides for a "user-defined" high limit which can be established for any assignment in the network. A warning is provided to the user, when that limit has been exceeded. Conversely, the low limits, also "user-defined", can be used to establish whether a billet node has enough officers assigned. The model once again will provide a warning to the user when the number of officers falls below that low limit. Specific instructions for establishing or changing these high and low limits will be explained later.

7. Data Files

There are five empirical or default data files.

a. Nodes

This file contains the number of officers assigned to each specific assignment in the model. Since

each assignment can vary in tour length,the number of officers is further broken down by the number of quarters they have left to serve at that specific assignment. For example, for department head school during the third tour there may be 75 officers with one quarter left and 75 officers with two quarters left to serve before being transferred elsewhere.

b. Arcs

The transfer percentages for each assignment in the network are contained in this file. For example, from department head school during the third tour, everyone proceeds to a fleet unit. Thus, the transfer percentages from "Third Tour Professional Training" to "Fourth Tour Fleet Unit" will be 100% and all other transfer percentages from "Third Tour Professional Training" will be 0%.

c. Length

Each assignment can range from 0 to 16 quarters long. The specific tour lengths for each assign ment are contained in this file. For example, for depart ment head school during the third tour the data file may list "Third Tour Professional Training" tour length as 2, meaning two quarters (6 months).

d. High Limits

Each assignment can have a "user- defined" high limit. The default high limits are contained in this file. For most assignments the default high limit is 9999.

e. Low Limits

Each assignment can have a "user- defined" low limit. The default low limits are contained in this file. For all assignments the default low limit is 0.

B. THE MODEL'S PROCEDURES

The model is capable of carrying out five major procedures: initialization of data; display of data; the calculation of the numbers of officers occupying assignments

in future years; changing data values; and saving data. These procedures are described below.

1. Initialization

The model variables' initial values are contained in files on the disk. The values for the number of officers in each assignment by quarters left, the transfer path percentages, the tour lengths, and the high and low limits are contained in formatted data files that are stored on the program disk. The user may also create his own data files and these will also be formatted and stored on the disk under file names chosen by the user. When the model is started up, the user must choose whether to initialize the variables with the default values, or another set of values he had previously saved. Each of the data types i.e. nodes, arcs, lengths, high and low limits, are formatted differently. The program prevents the user from selecting an improperly formatted data set during operation of the model. It also properly formats data that the user decides to save. Although the user could employ other means to read these data files, he should not do so, in order to avoid accidentally changing the format by unwittingly inserting hidden characters, and thus rendering the data file useless to the program.

2. Displays

The model offers several different types of data displays, each designed to highlight a specific type of data.

a. An Activity v.s. All Tours

This display shows the number of officers assigned to a generic activity for each tour. For example,

	FIRST	SECOND	THIRD ....
WASH DC	14	34	23

b. A Tour v.s. All Activities

This display shows the number of officers during a specific tour for all activities. For example,

EIGHTH TOUR

PROFESSIONAL TRAINING	0
PROFESSIONAL EDUCATION	6
WASHINGTON DC	25

etc...

c. Transfer Paths From a Billet

This display shows the percentages and the number of officers that would be transferred from a specific assignment to all possible next assignments. For example,

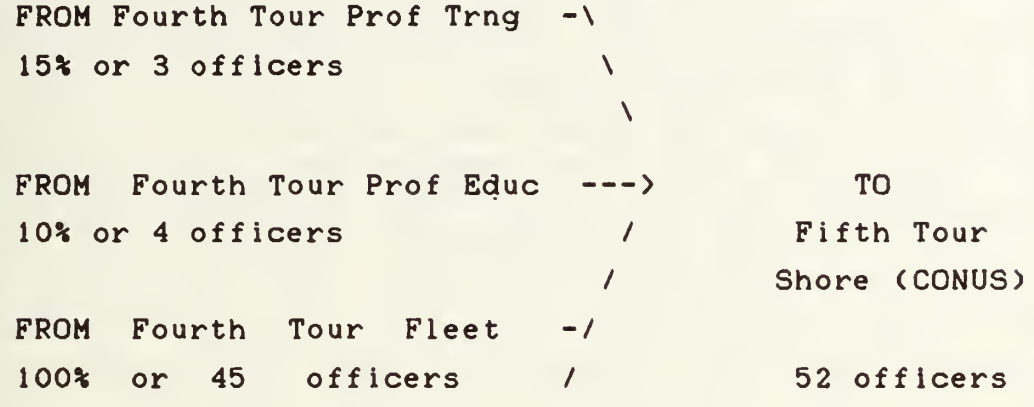
```
                  / Fourth Tour Prof Trng
                 /     0% or 0 officers
FROM            /
Third Tour  /     Fourth Tour Prof Educ
Prof Trng   \        0% or  0  officers
45 officers  \
              \  Fourth Tour Fleet
               \  100% or 45 officers

                    etc..
```

d. Transfer Paths To a Billet

This display shows the percentages and the number of officers that would be transferred from all possible previous tour assignments to a specific assignment. The percentages listed are the percentages of officers

transferred out of all the officers present in various assignments of the preceding tour. For example,



etc..

e. Billet Nodes by Quarters Left

This display shows the number of officers assigned to a specific assignment broken down by the number of quarters they have left to serve in that assignment. For example,

Third Tour Prof Trng

Officers Assigned:

75 with 1 quarter left

75 with 2 quarters left

0 with 3 quarters left

etc....

3. Calculations

The model manipulates all data by quarter and all data is also by quarter. For example, the number of

officers stationed at a specific assignment are identified by how many quarters they have left to serve at that assignment. The transfer percentages are the percentage of officers in their last quarter at their current assignment to be transferred each quarter to their next assignment. The maximum tour length allowed in the model is 4 years or 16 quarters. When the model asks for how many years and how many quarters to run the model, the program accepts the answer and converts it into quarters, then uses that number to determine the number of iterations the model should make. Upon completing the calculations, the number of quarters is converted back into number of years and quarters and displayed that way. Each quarter the performs a series of calculations.

a. First, for every assignment in the network, the numbers of officers with exactly one quarter left in their current assignments are placed in a temporary file.

b. Second, for every assignment in the network, all officers are shifted down to one fewer quarter. For example, if in an assignment that is four quarters long there were 15 officers with three quarters left, and 10 officers with four quarters left, after the shift there will be 15 officers with two quarters left, and 10 officers with three quarters left.

c. Third, the numbers of officers placed in the temporary file are multiplied by the appropriate transfer percentages and the resultant numbers are placed in the next assignments, each with the total number of quarters left appropriate for that assignment. For example, if there were 10 officers placed in the temporary file for the assignment "Third Tour Professional Training", the number 10 is multiplied by the transfer percentages for "Third Tour Professional Training", to determine how many of the 10 officers will be transferred along each transfer path to their next assignments. If the transfer percentages for

"Third Tour Professional Training" are 20% to "Fourth Tour Washington DC", 70% to "Fourth Tour Fleet", and 10% to "Afloat Staff", then 2 officers will be transferred to "Fourth Tour Washington DC", 7 to "Fourth Tour Fleet", and 1 to "Afloat Staff." If the tour length of "Fourth Tour Washington DC" is 3 years (12 quarters), the 2 officers transferred to that assignment will have 12 quarters left. If the tour length for the "Fourth Tour Fleet" is 2 years 2 quarters, the 7 officers transferred to that assignment will have 10 quarters left, and so on, for the other assignments.

d. Fourth, each quarter the program calculates two totals. The first total is the assignment total, i.e. the number of officers currently assigned to each particular assignment. The model adds up all officers assigned regardless of their quarters left in the assignment. The second total is the number of officers currently assigned to their "Xth" tour. This procedure sums up all officers currently serving their fourth tour, for example, in all activities.

e. Lastly, each quarter the program matches the assignment totals with the high and low limits. If either limit has been violated, the program provides a warning to the user (to be fully explained later).

4. Default Data Files

There are five different types of data sets used by the program. Each one is "formatted" or "coded" specifically for its own application, and there is no cross use of data sets. The program allows the user to designate his own data files and store data of his choice. These files are also specifically "formatted" for the data type and stored by the program. The five types of data sets used are:

a. Nodes Data

The default data file is called "Nodes" and contains the number of officers assigned to each assignment, by quarters left. If the user uses a directory program

(outside the model) to determine which files are on the program disk, all data files that contain nodes data are listed as having the filetype "nod". This filetype will not display during operation of the program, and the user should not declare it when naming files in the program. The program will automatically assign "nod" as the filetype on all nodes type data files.

b. Arcs Data

The default data file is called "Arcs" and contains the transfer percentage for each transfer path in the network. If the user uses a directory program to determine which files are on the program disk, all data files that contain arcs data are listed as having filetype "ard". This filetype will not display during operation of the program, and the user should not declare it when naming files in the program. The program will automatically assign "ard" as the filetype on all arcs type data files.

c. Length Data

The default data file is "Length" and contains the tour length, in quarters, for every assignment in the network. If the user uses a directory program to determine which files are on the program disk, all data files that contain length type data are listed as having filetype "led". This filetype will not display during operation of the program, and the user should not declare it when naming files in the program. The program will automatically assign "led" as the filetype on all length type data files.

d. High Limit Data

The default data file is called "Hilimit" and contains the upper constraints for the number of officers in each billet. If the user uses a directory program to determine which files are on the program disk, all data files that contain Hilimit data are listed as having filetype "hid". This filetype will not display during operation of the program, and the user should not declare it

when naming files in the program. The program will automatically assign "hid" as the filetype on all high limit type data files.

e. Low Limit Data

The default data file is called "Lolimit" and contains the lower constraints for the number of officers in each billet. If the user uses a directory program to determine which files are on the program disk, all data files that contain low limit data are listed as having filetype "lod". This filetype will not display during operation of the program, and the user should not declare it when naming files in the program. The program will automatically assign "lod" as the filetype on all low limit type data files.

5. Saving the Model's Data

When the user has completed a session with the model, he may desire to save the data currently in the model network, for instance, the current number of officers in each assignment, the current transfer path percentages, the current assignment tour lengths, or the current high or low limits. The model is capable of properly formatting each data type and saving it in one of three ways:

a. The user will be provided the opportunity to save each data type in a new file named by the user in which case the model will create a file using the name made up by the user, on the program disk, and transfer the current values of that data type into the new file. The data will automatically be properly formatted, thus allowing the user to recall this data for use in the future. For example, the user may wish to save the high limit data as a new file called "Newhigh," in which case the model will automatically assign that file the filetype "hid".

b. The user will be provided the opportunity to save each data type to a file already existing on the disk. The model will warn the user that saving data to an existing

file, will overwrite/destroy the previously saved data. For example, when the user chooses to save the high limit data to a file, the display will show the names of all the files on the disk that currently contain high limit data. If the user had previously created a file called "Newhigh", then this file would be listed. If the user now chose to save the current data to a file also called "Newhigh", then the old data in the file "Newhigh" would be overwritten/destroyed, and the data file "Newhigh" would now contain the current high limit data only.

c. The user will be also be provided the opportunity to replace the data in the default data file with the current data. The model will warn the user that replacing the default data file with the current data is a permanent step. The old default data is then lost, and when the program initializes with the default data, it will be with the "new" default data.

6. Changing Data

The model is also capable of changing the values of the various types of data the model uses, while the user is operating the model. To do so the user selects the option to change data from a menu, and by following the prompts can change the following data:

a. The Number of Accessions

The number of accessions into the surface warfare officer community. This is a single number that indicates the number of officers that have entered the Navy and have embarked on a career in the surface community. Since all prospective surface warfare officers begin their surface warfare careers at the Surface Warfare School (Basic), followed by their first tour at sea, the model uses the assignment "First Tour Fleet", to account for all accessions to the surface warfare officer community. This number remains the same every quarter (unless changed by the user), and new officers to the community only enter through

this assignment. The user can evaluate the impact of a greater or lesser number of accessions on the surface warfare officers career path by changing the number of accessions and using the model to make further computations.

b. The Transfer Path Percentages

The user may decide to evaluate the impact of changes to the path surface warfare officers take over the course of their career. One change might be to increase the number of officers that are ordered to professional education during their eighth tour. By changing the transfer path percentages from all activities in the seventh tour to increase the number of officers from those assignments to "Eighth Tour Professional Education", and then operating the model, the user can evaluate the effect that changes in the transfer paths percentages have on how many officers are ultimately transferred to the assignment "Eighth Tour Professional Education", and what impact this change has on tours nine through twelve. The increased number of officers being assigned to professional education in their eighth tour may decrease the officers available to be assigned to other eighth tour activities, and cause a gap of unfilled assignments in tours nine through twelve.

c. Tour Lengths

The user has the ability to alter the tour length of any assignment in the network. The user may choose to lengthen (up to a maximum of 16 quarters), or shorten a tour (down to a minimum of 0 quarters). If the user chooses to lengthen the tour from 4 quarters, to say, 6 quarters, the model assumes that those officers currently serving in the assignment will not have their tours extended to meet the new tour length but instead will leave as previously scheduled. Similarly, if the user chooses to shorten the tour, those officers currently serving in the assignment will continue to stay at that assignment until they have completed their original schedule.

d. High Limit Numbers

The user can change the high limit numbers for any assignment. This allows the user to set up a high limit constraint, run the model for "X" years and determine if the high limit constraint is ever violated. For example, in department head school, the physical seating capacity for students may be 150 seats. Of those students attending department head school, some are in their third tours and some in their fourth, e.g. 80% in their third tour, and 20% in their fourth tour. So 120 seats can be occupied by third tour officers and 30 seats can be occupied by fourth tour officers. These numbers may be used as the high limits that are imposed on the assignments, "Third Tour Professional Training" (120) and "Fourth Tour Professional Training" (30). If while running the model these high limits are exceeded, the model warns the user, and provides him with options (explained in detail later) from which to choose.

e. Low Limit Data

The user can also change the low limit numbers for any assignment in the network. Using the previous example of the department head school, the user may wish to make sure that at least 110 officers on their third tour and 15 officers on the fourth tour, are currently assigned to department head school. The user should change the low limit data for the "Third Tour Professional Training" to 110 and "Fourth Tour Professional Training" to 15. When the model makes the calculations and finds that the number of officers in the assignment "Third Tour Professional Training" falls below the low limit (110) the model warns the user and provides him with options (explained in detail later) from which to choose.

7. Reinitializing the Data

The user is also provided the opportunity to reinitialize some or all of the data types, i.e. nodes, arcs, length, hilimit, and lolimit. For example, after

running the model for three years and not violating a low limit for "Third Tour Professional Training" (110), the user may wish to change the low limit to 115 officers, and run the model again for three years, to see if this new low limit will be violated. Without reinitializing the model's data, the model would commence running with three years calculations already completed, and the result would be a total of six years calculations into the future. Instead, the user may desire to reinitialize the nodes data, and any other data type of his choosing, and then run the model for three years. One word of warning is necessary here. If the user chose to change the low limit from 110 to 115 officers, and then decided to reinitialize the low limit data, the net result would be to destroy the change the user had made to the low limit data, because the low limit for "Third Tour Professional Training" would also be reset to the value found in the initial low limit data file. For that reason, the user is provided the opportunity to reinitialize separately each of the following data types: the nodes data, the arcs data, the length data, the high limit data, and the low limit data.

III. PROCEDURES FOR USING MODEL

A. THE DISTRIBUTION DISKS

The model package contains four disks. Two are labeled "IBM Compatible", Distribution Disks I and II, and two are labeled "Heath/Zenith 100" Distribution Disks I and II. This model was written in two versions, one for use on any IBM-PC compatible and one for use on a Heath/Zenith 100 micro computer. Both versions require MSDOS Version 2.0 or the equivalent. Since each requires a different operating system, they are not interchangeable between micro computers with different operating systems. Color has been used in the displays to enhance the prompts, but the program will still run using a black and white monitor, with no degradation. Distribution Disk I contains all the files necessary for the program to run, including the main program, plus the six default data files. The following is a list of the files contained on Distribution Disk I:

SWOPATH.COM
NODES.NOD
NODEZERO.NOD
ARCS.ARD
LENGTH.LED
HILIMIT.HID
LOLIMIT.LID

Note: After operating the program, the user may have created one or more additional data files, that will also be listed on the directory of the "working copy" of Distribution Disk I.

Distribution Disk II contains the source code listings for the model's main program for those users interested in performing possible future changes to the program. . These listings are also printed in Appendices A through N. This

disk also includes a program called UPDATE.COM that allows the user to reenter the assignment data in the NODES.NOD default data file. The following is a list of the files contained on Distribution Disk II:

SWOPATH.PAS
ZENUTILS.PAS (only on Heath/Zenith disks)
IBMUTILS.PAS (only on IBM Compatible disks)
LOGO.PAS
INITDATA.PAS
SELECTIO.PAS
CALCULAT.PAS
TOTALS.PAS
CHGDATA.PAS
SCREENS.PAS
DISPLAYS.PAS
DATADUMP.PAS
ANSWERS.PAS
STORDATA.PAS
UPDATE.COM (listing not included)

B. GETTING STARTED

The user should select the two distribution disks labeled for use on his micro computer (Either the "IBM Compatible" disks, or the "Heath/Zenith 100" disks.). The distribution disks are not copy-protected for the user's convenience. The user is reminded that this product is U.S. Government property and all appropriate rules for such apply. The DISKCOPY command should be used to make exact copies ("working copies") of each distribution disk. The original distribution disks should be kept separately in a safe place. The working copy of Distribution Disk I should now be placed into drive A.

C. SAMPLE PROGRAM EXECUTION

The following will be a detailed explanation of how the program works, including actual displays (some slightly modified, due to space considerations) that the user can expect to see, if the user follows the procedures described in the text. Entries required to be made by the user are indicated within quotation marks. For example, if the user is required to press the letter A, it will be indicated in the text as "A", if the user is required to press the number 8 and then press carriage return (enter, or return), this will be indicated in the text as "8<CR>". All character entries will be indicated in the text by upper case letters. It is not necessary for the user to use upper case letters when making entries since the program will recognize either upper or lower case responses as correct.

1. Selecting Input Data Files

To start the program, type "SWOPATH<CR>" at the prompt: A>. The first screen will be the logo including the version number of the program, which will automatically change to the display depicted in Figure 3.1 below.

Do you want to change the input data files from those listed below?

Data:	Data files:
0. No changes wanted/finished changes.	
1. Number of officers at each assignment:	Nodes
2. Transfer path percentages:	Arcs
3. Assignment tour lengths:	Length
4. High limits:	Hilimit
5. Low limits:	Lolimit

Type your selection [0,1,2,3,4,5]:

Figure 3.1 Input Data File Screen

This screen provides the user with the option of utilizing the default data files listed, or changing to a data file of the user's choice. If the user chooses something other than 0,1,2,3,4,or 5, the program provides

the brief message near the bottom of the screen, 'Your response is incorrect, please try again', and returns to the original screen shown in Figure 3.1 above. As an example, let's choose to change the data file for the number of officers at each assignment by pressing "1". Figure 3.2 is the display that now is on the screen.

Remember, you must choose a data file that you have previously saved. Those are listed below.

NODEZERO
NODES

Enter the input filename
typed exactly as listed above,
that you choose to use:
Type NODES if you want to exit back to menu.

Figure 3.2 Change Data File Screen

The two files 'NODEZERO and NODES' are the only files, currently on the distribution disk, that contain the formatted data indicating the number of officers at each assignment. If the user wanted to return to the previous menu, Figure 3.1 without making any changes he would follow the instruction on the last line of the display in this case typing the word NODES. If the user types a filename other than one of those listed, the program provides a brief error message and returns to the screen in Figure 3.2. If the user has earlier created his own assignment data files, they would be listed along with the NODEZERO and NODES files.

If the user had chosen "2" from the screen shown by Figure 3.1 the next screen display would be similar to that in Figure 3.2, but would instead list the files containing transfer path percentage data. Similar statements may be made for choices 3, 4 or 5.

In this example we will select NODES by entering "NODES<CR>". The program now returns to the screen in

Figure 3.1, to provide the user the opportunity to make another change in the input data files. After all desired changes have been made, typing "0" will so indicate to the program which will then respond by printing across the bottom of the screen, 'Initializing . . . . . '. The program is now reading each data file into memory storage. Each time it reads a data file into memory, it informs the user by printing the initializing message. Once it completes reading in all five data files, the program automatically proceeds to the next display shown in Figure 3.3.

Currently the number of accessions each quarter is
350 officers
The new number of accessions each quarter is:
Press <CR> for no change.

Figure 3.3 Change Number of Officer Accessions Screen

2. Initializing the Number of Officer Accessions

This screen provides the user with the opportunity to initialize the number of officer accessions used by the program. The default value for officer accessions per quarter is 350 officers. The user can either type a new number or by just pressing the carriage return,"<CR>", the user accepts the the default number of officer accessions and the program automatically displays the next screen, Figure 3.4.

3. The Selection Menu

This screen provides the user with the primary options he will require to operate the program. From this selection menu the user can display data, change data, reinitialize data values or conduct calculations. Note at the top of the screen that the program tells the user for how many years and quarters the calculations have been

completed. No calculations have been performed yet, so the indication is 0 years and 0 quarters. So far the only operation the program has conducted was to read in the data to fill the network with the required values.

Calculations have been completed for 0 years and 0 quarter

Choose one of the following selections.

0. All finished.
1. Review the display selections.
2. Display an activity v.s. all tours
3. Display a tour v.s. all activities
4. Display the transfer paths FROM an assignment
5. Display the transfer paths TO an assignment
6. Display the assignments by quarters left
7. Change data values
8. Reinitialize data values.
9. Ready for calculations.

Please type in the number of your selection
[0,1,2,3,4,5,6,7,8,9]

Figure 3.4 Selection Menu Screen

4. Data Displays

a. Activity v.s. All Tours

By pressing "2" to display an activity v.s. all tours the program first provides the user with a menu (see Figure 3.5) from which to choose the activity the user wishes to see.

Which activity are you interested in?

A. Professional Training
B. Professional Education
C. Washington DC
D. Shore (CONUS)
E. Afloat Staff
G. Shore (OUTUS)
H. Separation

Type your choice:

Figure 3.5 Display Activity v.s. Tour Selection Screen

When selecting Shore (CONUS) by pressing "D" the program indicates the number of years and quarters for which calculations have been completed. It indicates that the

user chose the activity, Shore (CONUS), and then provides a breakdown of the number of officers serving in a Shore (CONUS) assignment during their first tour, second tour, etc.. (See Figure 3.6) For instance, there are 204 officers currently in their fourth tour, serving in an assignment that falls under the general activity, Shore (CONUS). The program then provides the user with the opportunity to display another activity breakout. By answering 'Y' the user would see the screen depicted in Figure 3.5 above, and have the opportunity to choose a different "activity v.s. tour" breakout. If the user desires to display no other activities he types "N".

For 0 years and 0 quarter(s) calculations.

For: Shore (CONUS)

TOURS.....

FIRST	SECOND	THIRD	FOURTH	FIFTH	SIXTH
0	780	270	204	204	300
SEVENTH	EIGHTH	NINTH	TENTH	ELEVENTH	TWELFTH
300	300	350	200	300	0

Do you desire to see another activity breakout? (Y/N)

Figure 3.6 Activity v.s. Tour Breakout Screen

The user is then returned back to the Selection Menu shown in Figure 3.4 above.

b. Tour v.s. All Activities

By pressing "3", the program first provides the user with a menu (Figure 3.7) in order to choose which tour the user wishes to examine. Note that the display prompts the user to type the number of his choice and then to press the carriage return key <CR>.

When selecting the Fourth Tour by entering "4<CR>", the program again indicates for how many years and quarters calculations have been completed. The display also

indicates that the user chose the Fourth Tour, and then provides a breakdown of the total number of officers who are

```
Which tour do you wish to see displayed?
 TOUR

 1.  FIRST
 2.  SECOND
 3.  THIRD
 4.  FOURTH
 5.  FIFTH
 6.  SIXTH
 7.  SEVENTH
 8.  EIGHTH
 9.  NINTH
10.  TENTH
11.  ELEVENTH
12.  TWELFTH

TOUR:        (type number<CR>)
```

Figure 3.7 Display Tour v.s. Activity Selection Screen

in their Fourth Tour by assignments categorized as Professional Training, Professional Education, Washington DC, Shore (CONUS), etc.. (see Figure 3.8) For instance,

```
For 0 years and 0 quarter(s) calculations.
        For: Fourth Tour
Activity:                          Number of officers
Professional Training      :                50
Professional Education     :               200
Washington DC              :               168
Shore (CONUS)              :               204
Fleet Unit                 :               480
Afloat Staff               :                96
Shore (OUTUS)              :                24
Separation                 :                 0
Total Officers             :              1222
Do you desire to see another tour breakout? (Y/N)
```

Figure 3.8 Tour v.s. Activity Screen

there are 204 officers currently in their Fourth Tour, serving in an assignment that falls under the general activity, Shore (CONUS). The program then provides the user with the opportunity to display another tour breakout. By answering 'Y' the user would see the screen depicted in Figure 3.7 above, and have the opportunity to choose a different "tour v.s. activity" breakout. By typing "N" the user is returned back to the Selection Menu shown in Figure 3.4 above.

c. Transfer Paths From an Assignment

By pressing "4" to see a display for the transfer paths from an assignment the program will provide

---

Which assignment do you wish to see the transfer paths from?

TOUR		ACTIVITY	
1.	FIRST	A.	PROFESSIONAL TRNG
2.	SECOND	B.	PROFESSIONAL EDUC
3.	THIRD	C.	WASHINGTON DC
4.	FOURTH	D.	SHORE CONUS
5.	FIFTH	E.	FLEET UNIT
6.	SIXTH	F.	AFLOAT STAFF
7.	SEVENTH	G.	SHORE OUTUS
8.	EIGHTH	H.	SEPARATION
9.	NINTH		
10.	TENTH		
11.	ELEVENTH		
12.	TWELFTH		

TOUR: (type number<CR>) ACTIVITY: (type letter)

Figure 3.9 Display Transfer Paths FROM Selection Screen

the user with a menu (see Figure 3.9) to select which assignment the user wants to see the transfer path percentages FROM. Note that once again the display prompts the user to type a number and then press the carriage return <CR>. If the user selects either TWELFTH TOUR or the SEPARATION activity, the program will briefly warn the user that there are no transfer paths FROM either of these two options, and then allows the user to change his selection.

Pressing "3<CR>" and then "F" to select the THIRD TOUR, AFLOAT STAFF assignment. Figure 3.10 will appear on the next screen.

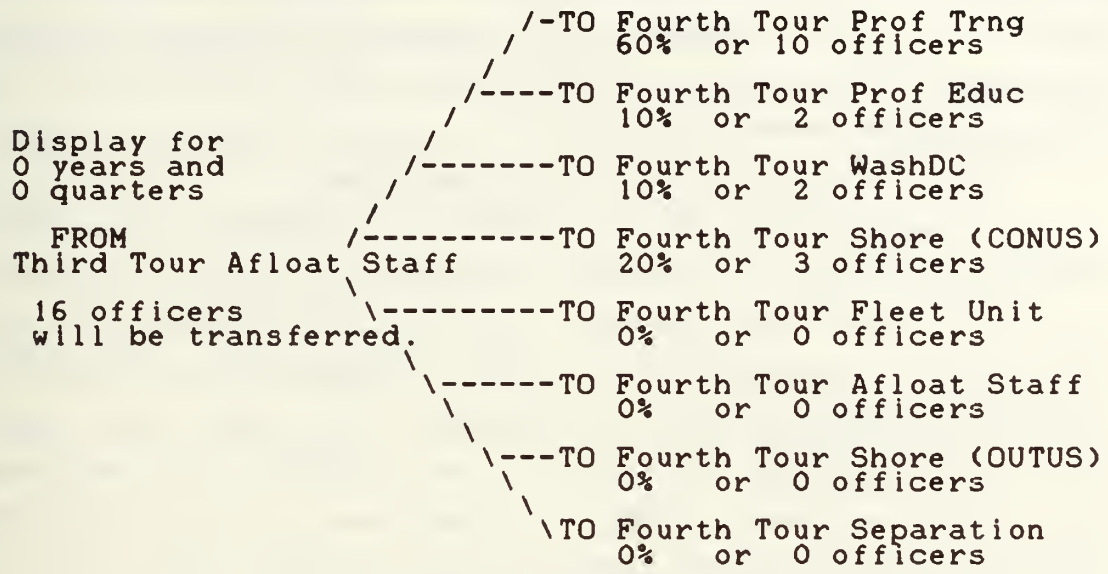


Figure 3.10 Transfer Path Percentages FROM Screen

Once again the program indicates for how many years and quarters calculations have been completed and indicates assignment the user has selected to see the transfer path percentages FROM. The display shows how many officers will be transferred from the Third Tour Afloat Staff assignment and then on the right, indicates where those officers will be transferred. In Figure 3.10 the display shows that 16 officers will be transferred. This means that at the Third Tour Afloat Staff assignment there are 16 officers with one quarter left to serve. When the program begins calculations these 16 officers will be transferred as shown in the figure. Namely, sixty percent, or 10 of those 16 officers will be transferred to a Fourth Tour Professional Training assignment, ten percent or 2 officers will be transferred to a Fourth Tour Professional

Education assignment, etc.. The individual numbers may not sum up to the total officers transferred due to rounding.

Once again the program provides the user with the opportunity to choose to see another transfer path breakout. If the user chooses 'Y', then the screen shown in Figure 3.9 will appear and the user may again select an assignment. By pressing "N" the program will return to the Selection Menu shown in Figure 3.4.

d. Transfer Paths To an Assignment

Pressing "5" to display the transfer paths TO an assignment, Figure 3.11 shows the screen the user will see.

On this screen the program provides the user with a menu for selecting to which assignment the user wants to see the transfer path percentages. Note that once again the display prompts the user to type a number and then press the carriage return <CR>. If the user selects FIRST TOUR the program will briefly warn the user that there are no

---

Which assignment do you wish to see
the transfer paths to?

TOUR		ACTIVITY	
1.	FIRST	A.	PROFESSIONAL TRNG
2.	SECOND	B.	PROFESSIONAL EDUC
3.	THIRD	C.	WASHINGTON DC
4.	FOURTH	D.	SHORE CONUS
5.	FIFTH	E.	FLEET UNIT
6.	SIXTH	F.	AFLOAT STAFF
7.	SEVENTH	G.	SHORE OUTUS
8.	EIGHTH	H.	SEPARATION
9.	NINTH		
10.	TENTH		
11.	ELEVENTH		
12.	TWELFTH		

TOUR: (type number<CR>) ACTIVITY: (type letter)

Figure 3.11 Display Transfer Paths TO Selection Screen

transfer paths TO that option, and then allows the user to change his selection. Pressing "6<CR>" and then "E" to

select the SIXTH TOUR, FLEET UNIT assignment, a screen showing Figure 3.12 will appear next.

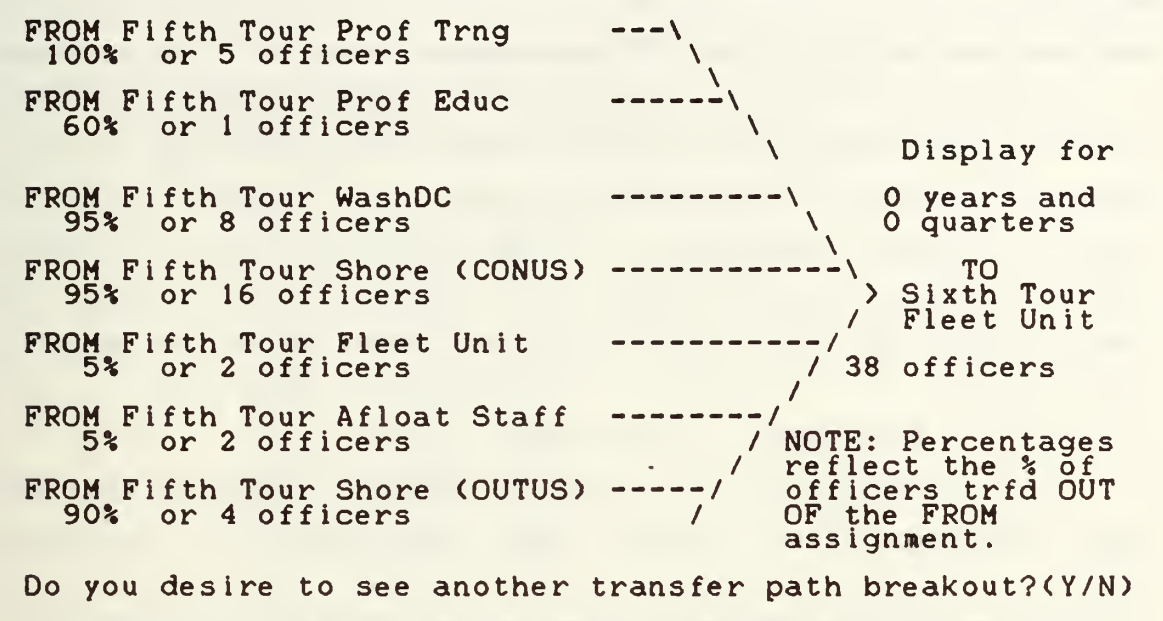


Figure 3.12 Transfer Path Percentages TO Screen

Once again the display indicates the number of years and quarters for which calculations have been completed, and indicates the tour, Sixth Tour Fleet Unit, that the user selected to see the transfer path percentages TO. This display allows the user to determine where the officers being transferred into the assignment he is interested in are coming from. The information on the left side of this screen is the percent of the total officers in their last quarter at the indicated FROM assignment that will be transferred to the selected TO assignment. The number following the percent figure in each row is the actual number of officers to be transferred FROM the indicated assignment in each row on the left TO the assignment on the right side of the screen. For instance, there will be 16 officers transferred from Fifth Tour Shore (CONUS) to Sixth Tour Fleet Unit. The 16 officers are ninety five percent of all the officers assigned to Fifth

Tour Shore (CONUS) with one quarter left. Occasionally there will be a percent indicated, yet 0 officers shown to be transferred. This occurs when there are no officers with one quarter left at the FROM assignment. Also the numbers of officers on the left may not sum to the number indicated as the total transferred due to rounding. Choosing 'Y' will return the user to the display shown in Figure 3.11, to select another assignment to which he may want to display the transfer path percentages. If the user is finished, he would press "N" and thereby return to the Selection Menu of Figure 3.4.

e. Assignment by Quarters Left

Figure 3.13 shows the screen the user will see when pressing "6" to display the assignments by quarters left.

```
Which assignment do you wish to see?

TOUR                    ACTIVITY

 1.  FIRST              A.  PROFESSIONAL TRNG
 2.  SECOND             B.  PROFESSIONAL EDUC
 3.  THIRD              C.  WASHINGTON DC
 4.  FOURTH             D.  SHORE CONUS
 5.  FIFTH              E.  FLEET UNIT
 6.  SIXTH              F.  AFLOAT STAFF
 7.  SEVENTH            G.  SHORE OUTUS
 8.  EIGHTH             H.  SEPARATION
 9.  NINTH
10.  TENTH
11.  ELEVENTH
12.  TWELFTH

TOUR:   (type number<CR>)  ACTIVITY:    (type letter)
```

Figure 3.13 Display Assignments by Quarters Left Selection Screen

The program here provides the user with a menu to select which assignment from which the user wants to see the breakout of officers assigned by quarters left. Again the display prompts the user to type a number and then press the carriage return <CR>.

Pressing "9<CR>" and then "G" to select the NINTH TOUR, SHORE OUTUS assignment Figure 3.14 will appear on the next screen.

```
After 0 year(s) and 0 quarter(s)
     Ninth Tour Shore (OUTUS)
     Tour Length of 12 quarters.
Officers assigned:

   5        with  1 qtr  left
   5        with  2 qtrs left
   5        with  3 qtrs left
   5        with  4 qtrs left
   5        with  5 qtrs left
   5        with  6 qtrs left
   5        with  7 qtrs left
   5        with  8 qtrs left
   5        with  9 qtrs left
   5        with 10 qtrs left
   5        with 11 qtrs left
   5        with 12 qtrs left
   0        with 13 qtrs left
   0        with 14 qtrs left
   0        with 15 qtrs left
   0        with 16 qtrs left

 The total number of officers assigned is 60
Do you desire to see another assignment?(Y/N)
```

Figure 3.14 Officers Assigned by Quarters Left Screen

Here, the program indicates the years and quarters for which calculations have been completed and indicates the assignment selected by the user, Ninth Tour Shore (OUTUS). This display also indicates the tour length of the assignment. As discussed previously, the maximum length of any assignment in the model is 16 quarters. Although not all assignments are for 16 quarters, the display will always show 16 quarters of information. Note that there are 0 officers assigned with 13, 14, 15, and 16 quarters left.

The display also indicates the total number of officers assigned to the selected assignment.

As before the program provides an opportunity for the use to choose to see another display of officers assigned by quarters left. If finished, pressing "N" returns the user to the Selection Menu of Figure 3.4.

This completes the examples of the different types of displays available with the program. If a less experienced user desires to review the displays, selection number 1 of the Selection Menu of Figure 3.4 will provide him with a brief review of each of the available displays.

5. Changing Data

In order to change data values, the user must press "7" from the list on the Selection Menu of Figure 3.4. Figure 3.15 shows the next screen the user can expect to see.

```
Which data do you desire to change?

Input data                                  Data file

0.  No changes/finished changes.
1.  Number of officer accessions.
2.  Transfer path percentages:              Arcs
3.  Assignment tour lengths:                Length
4.  High limits:                            Hilimit
5.  Low limits:                             Lolimit

Type your selection [0,1,2,3,4,5]:
```

Figure 3.15 Data Change Menu Screen

Here, the program provides the user with the opportunity to change all the data in the program with the exception of the number of officers presently assigned to each billet. In addition to listing the possible categories of input data the user might wish to change, the display also indicates the data file the program is presently employing to establish that data, where applicable. The menu in Figure 3.15 allows the user to select the data he desires to change.

a. Changing the Number of Accessions

Pressing "1" to select the number of officer accessions the screen will show Figure 3.16.

Currently the number of accessions each quarter is
350 officers.

The new number of accessions each quarter is:
Press <CR> for no change.

Figure 3.16 Changing Number of Officer Accessions Screen

This display shows the number of officers entering the Surface Warfare Officer Community from all sources, each quarter. The program provides the user an opportunity to change this number by simply typing the new number and pressing the carriage return. If the user does not want to change the value then pressing the carriage return "<CR>"keeps the number of officer accessions at 350 officers. The program displays the new number of officer accessions as selected by the user and then returns to the change menu shown in Figure 3.15.

Which assignment do you wish to change the path % from?

TOUR		ACTIVITY	
1.	FIRST	A.	PROFESSIONAL TRNG
2.	SECOND	B.	PROFESSIONAL EDUC
3.	THIRD	C.	WASHINGTON DC
4.	FOURTH	D.	SHORE CONUS
5.	FIFTH	E.	FLEET UNIT
6.	SIXTH	F.	AFLOAT STAFF
7.	SEVENTH	G.	SHORE OUTUS
8.	EIGHTH	H.	SEPARATION
9.	NINTH		
10.	TENTH		
11.	ELEVENTH		
12.	TWELFTH		

TOUR: (type number<CR>) ACTIVITY: (type letter)

Figure 3.17 Change Transfer Path Percentages Selection Screen

b. Changing the Transfer Path Percentages

Pressing "2" to select transfer path percentages the screen will show Figure 3.17.

The program here provides the user with a menu to select from which assignment the user wants to change the transfer path percentages.

Pressing "3<CR>" and the "E" to select the THIRD TOUR FLEET UNIT assignment the screen will show Figure 3.18.

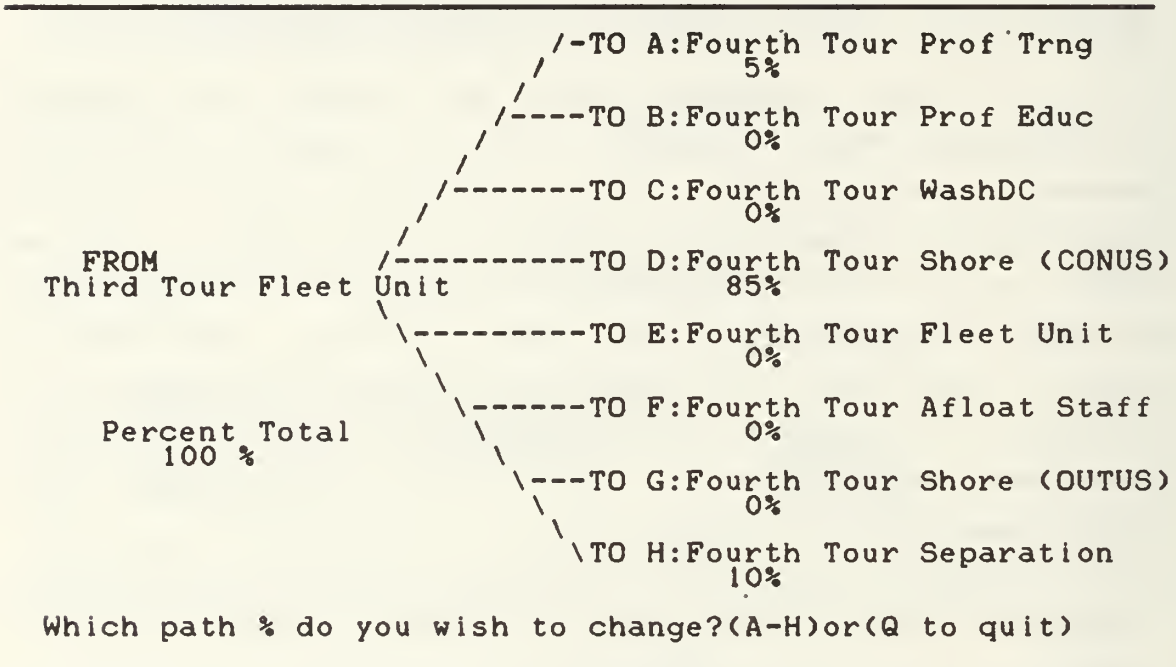


Figure 3.18 Change Transfer Path Percentages Screen

This display shows the current transfer path percentages from the indicated assignment to each assignment in the next tour. Note the program prompt at the bottom of the screen. Selecting "A" the cursor will move to the line below "TO A: Fourth Tour Prof Trng" just before the 5%. To change the transfer percentage, simply type the new value and press the carriage return. For example, typing "30<CR>" the display will change in two places. Where before the percentage read 5% in normal video, it now reads 30% in reverse video. The reverse video feature is to indicate to the user that he has changed that particular value. The second display change is where the percent total is indicated. The number displayed will be 125%, displayed in

reverse video to indicate to the user that a change has occurred. The reverse video feature will occur whether the user changes the percentage or simply types the exact same percentage again. After making the above change, the cursor has now returned to the bottom of the screen, prompting another change.

If the user now types "Q" to quit this section of the program a warning will be displayed at the bottom of the screen to indicate to the user that he cannot leave this section of the program, until he forces the percent total to equal either zero, or one hundred percent. The user In the present case the user is warned that the percent total does not equal either zero or one hundred, the current percent total (here 125%) is displayed, and he is instructed to continue to make changes to the transfer path percentages until one of the mentioned totals is achieved. The user is then returned to the original prompt and asked which path percentage the user desires to change. This time pressing "D" to change the transfer percentage to Fourth Tour Shore (CONUS) the cursor has moved to immediately before the 85%. Typing "60<CR>", the percentage should now be highlighted in reverse video, and the percent total should now read one hundred percent. The cursor also returns to the bottom of the screen, to prompt the user for another change. Now that the 100 percent total has been achieved, typing "Q" allows the user to leave this section by asking him if he desires to change another transfer path percentage. Typing "N", the program automatically returns to the change menu of Figure 3.15.

c. Changing the Assignment Tour Lengths

Pressing "3" for assignment tour lengths, the screen will show Figure 3.19.

```
Which assignment tour length do you wish to change?

TOUR                    ACTIVITY

 1.  FIRST              A.  PROFESSIONAL TRNG
 2.  SECOND             B.  PROFESSIONAL EDUC
 3.  THIRD              C.  WASHINGTON DC
 4.  FOURTH             D.  SHORE CONUS
 5.  FIFTH              E.  FLEET UNIT
 6.  SIXTH              F.  AFLOAT STAFF
 7.  SEVENTH            G.  SHORE OUTUS
 8.  EIGHTH             H.  SEPARATION
 9.  NINTH
10.  TENTH
11.  ELEVENTH
12.  TWELFTH

TOUR:   (type number<CR>)  ACTIVITY:    (type letter)
```

Figure 3.19 Change Assignments Tour Length Selection Screen

Here the program provides the user with a menu to select the assignment for which the user wants to change the tour length. Figure 3.21 shows the next screen the user will see when pressing "3<CR>" and then "E" to select the THIRD TOUR, FLEET UNIT assignment.

```
The old Third Tour Fleet Unit length is 8 quarters
How many quarters long do you want assignment now?
```

Figure 3.20 Change Assignment Tour Length Screen

This display indicates the old tour length of 8 quarters for the Third Tour Fleet Unit and prompts the user to enter the new tour length in quarters. Caution: The user must enter a value and then press the carriage return. If he only presses the carriage return, without entering a value, the program may not operate properly. For example, typing "10<CR>" the program displays the old value and the new value, and then automatically returns to the change menu of Figure 3.15.

d. Changing the High Limits

Figure 3.21 shows the next screen the user will see if "4" is pressed for high limits.

```
Which high limit do you wish to change?

TOUR                    ACTIVITY

 1.  FIRST              A.  PROFESSIONAL TRNG
 2.  SECOND             B.  PROFESSIONAL EDUC
 3.  THIRD              C.  WASHINGTON DC
 4.  FOURTH             D.  SHORE CONUS
 5.  FIFTH              E.  FLEET UNIT
 6.  SIXTH              F.  AFLOAT STAFF
 7.  SEVENTH            G.  SHORE OUTUS
 8.  EIGHTH             H.  SEPARATION
 9.  NINTH
10.  TENTH
11.  ELEVENTH
12.  TWELFTH

TOUR:   (type number<CR>)  ACTIVITY:    (type letter)
```

Figure 3.21 Change High Limits Selection Screen

The program here provides the user with a menu to select the assignment for which the user wants to change the high limit. Figure 3.22 shows the next screen the user will see if pressing "3<CR>" and the "A" to select the THIRD TOUR, PROFESSIONAL TRNG assignment.

```
The old high limit for Third Tour Prof Trng
                                        is 9999 officers.
    There currently are   150   assigned.
How many officers do you want as the high limit now?
```

Figure 3.22 Change High Limit Display

This display indicates the old high limit for the Third Tour Prof Trng, the number of officers currently assigned to the Third Tour Prof Trng, and prompts the user to enter the new high limit. The user is reminded here that the default setting for all high limits in the program is 9999. Caution: The user must enter a value and then press the carriage return. If the user only presses the carriage return, without entering a value, the program may not operate properly. For example, typing "350<CR>", the program will display the old value and the new value, and

then asks the user if he wants to change another high limit. In order to prepare for the example in Section 5, another high limit choice is made by first typing "Y" and at the High Limits Selection Screen (Figure 3.21), FOURTH TOUR, PROFESSIONAL TRNG is selected by typing "4<CR>" and "A". At the prompt on the High Limit Change Display (Figure 3.22), "150<CR>" is typed. Then to finish with this section, typing "N" at the prompt asking if another change is desired will automatically return the user to the change menu of Figure 3.15.

e. Changing the Low Limits

Figure 3.23 shows the next screen the user will see if "5" is pressed for low limits.

Which low limit do you wish to change?

TOUR		ACTIVITY	
1.	FIRST	A.	PROFESSIONAL TRNG
2.	SECOND	B.	PROFESSIONAL EDUC
3.	THIRD	C.	WASHINGTON DC
4.	FOURTH	D.	SHORE CONUS
5.	FIFTH	E.	FLEET UNIT
6.	SIXTH	F.	AFLOAT STAFF
7.	SEVENTH	G.	SHORE OUTUS
8.	EIGHTH	H.	SEPARATION
9.	NINTH		
10.	TENTH		
11.	ELEVENTH		
12.	TWELFTH		

TOUR: (type number<CR>) ACTIVITY: (type letter)

Figure 3.23 Change Low Limits Selection Screen

The program provides the user with a menu to select the assignment for which the user wants to change the low limit. Figure 3.24 shows the next screen the user will see when pressing "3<CR>" and then "E" to select the THIRD TOUR, FLEET UNIT assignment.

```
The old low limit for Third Tour Prof Trng is 0 officers.

    There currently are    80    assigned.

How many officers do you want as the low limit now?
```

Figure 3.24 Change Low Limit Screen

This display indicates the old low limit for the Third Tour Fleet Unit, the number of officers currently assigned to the Third Tour Fleet Unit, and prompts the user to enter the new low limit. The user is reminded here that the default setting for all low limits in the program is 0. Caution: The user must enter a value and then press the carriage return. If the user only presses the carriage return, without entering a value, the program may not operate properly. For example, typing "0<CR>" the program displays the old value and the new value, and then asks the user if he wants to change another low limit. Then to finish with this section "N" is typed at the prompt asking if another change is desired. The program automatically returns to the change menu of Figure 3.15.

6. Reinitializing the Data

Pressing "0" indicates that the user is finished making changes. The program then automatically returns to the Selection Menu of Figure 3.4.

```
Which data do you desire to reinitialize?

Data:                                              Data files:

0.  None/finished reinitializing data.
1.  Number of officers at each assignment:         Nodes
2.  Transfer path percentages:                     Arcs
3.  Assignment tour lengths:                       Length
4.  High limits:                                   Hilimit
5.  Low limits:                                    Lolimit

Type your selection [0,1,2,3,4,5]:
```

Figure 3.25 Reinitialize Data Selection Screen

To reinitialize data values, the user may press "8". Figure 3.25 shows the next screen the user will see.

The program allows the user to reinitialize all data used in the program. The display above shows the data and the corresponding data file currently being used by the program. The user simply types the number (1 through 5) of his selection, and the program will initialize that data type, using the file listed. The program also sends the user a message along the bottom of the screen to let the user know that the program is in fact initializing the data selected. A word of caution is required here. If the user has been operating the program, and has made changes to the data, as for example the high limits for two assignments have been changed in Section 5.d., then the program will destroy all such changes by replacing the changed data with the data file listed above when reinitializing the same data type (High limits in this case).

Pressing "2", for example, to reinitialize the transfer path percentages, note the message that appears along the bottom of the screen, indicating the program is initializing the transfer path data using the file Arcs. After reinitializing the selected data, the program returns the user to the Reinitializing Data Selection Screen (Figure 3.25) for another selection by the user. When finished, pressing "0" will so indicate to the program. The program then automatically returns the user to the Selection Menu of Figure 3.4.

7. Calculations

Note that this display still indicates that 0 year(s) and 0 quarter(s) calculations have been completed. By pressing "9" the user may select to commence calculations. At that point a brief message appears at the bottom of the screen instructing the user that if he wants to quit and return to the selection menu, he should choose 0

years and 0 quarters. The next question the program asks is if the user desires to reinitialize (set to zero) the years and quarters count. This will be explained further later in Section 7. For now pressing "N" the program asks how many years and quarters the user desires to run the model. As an example the user may enter "3<CR>" for the years and "2<CR>" for the quarters.

The program then commences calculations. It begins by taking those officers with 1 quarter left in each assignment and placing them in a temporary transfer file. It then subtracts one quarter from the number of quarters that all other officers have left in their current assignments. This way officers who earlier had 2 quarters left, will now have 1 quarter left, those with 3 quarters left to 2 quarters left, and so on. This process is carried our for each assignment in the network. The program then takes the number of officers in these temporary transfer files, multiplies them by the respective transfer path percentages, and adds the resultant number to the assignment they are being transferred to. For instance, if at the beginning of calculations there were 10 officers with one quarter left, 20 officers with two quarters left, and 30 with three quarters left, at Third Tour Fleet Unit then the 10 officers with one quarter left are placed in a temporary transfer file, and the 20 officers with two quarters left will have only one quarter left, while the 30 officers with three quarters left will have only two quarters left. The program then takes the transfer path percentages, for example thirty percent to Fourth Tour Professional Education and seventy percent to Fourth Tour Fleet Unit and multiplies the 10 officers being transferred from Third Tour Fleet Unit by thirty and seventy percent respectively. The resulting 3 and 7 officers are added to Fourth Tour Professional Education and Fourth Tour Fleet Unit. If, for example, the

tour length of Fourth Tour Professional Education is eight quarters and the tour length of Fourth Tour Fleet Unit is five quarters, the 3 officers will be added to the assignment Fourth Tour Professional Education with eight quarters left, and the 7 officers will be added to the assignment Fourth Tour Fleet Unit with five quarters left. One must remember that these 3 and 7 officers will not be the only officers added to those assignments. These are just the officers transferred from the assignment Third Tour Fleet Unit. There are probably several more officers being transferred from the other assignments in the Third Tour, to the Fourth Tour assignments. The results of these calculations are maintained by the program memory in temporary files, until all calculations are completed, at which time the data is transferred into the assignment data variables. If a limit either high or low, is violated, this transfer from temporary file to the assignment data variables does not take place and a warning is posted to the user as explained in the next section. The program maintains the assignment data as configured after the last successful calculations were completed.

a. Violating the Limits

After completing the transfer of all the officers to an assignment, the program checks to ensure that the high or low limits for that assignment have not been violated. If a limit is violated, the program sends the user a warning. As an example, Figure 3.26 shows the screen the user sees when after conducting calculations for 2 quarters, the constraint for Fourth Tour Professional Training is exceeded.

Earlier, the user created a high limit of 150 officers for Fourth Tour Professional Training. This limit was exceeded after only two quarters of calculations. The program now officers the user three options.

```
After 0 year(s) and 2 quarter(s)
The constraint for Fourth Tour Prof Trng has been exceeded
There are 152 officers there now.
The high limit is 150
What do you desire to do now?
0.  Abort simulation.  Return to selection menu.
1.  Ignore limits and continue simulation.
2.  Back up one quarter in the simulation.
```

Figure 3.26 Constraint Violation Screen

(1) Aborting the Simulation. The user can abort the simulation and return to the selection menu. The display of the selection will now indicate that 0 years and 0 quarters of calculations have been completed. When the user chooses to abort the simulation the program automatically returns the user to where the user was before he started the calculations that resulted in a violation of the limits. In this case the user started with 0 years and 0 quarters of calculations completed and therefore the program returned the user to that status. If for example, the user had first operated the program for 3 years without violating a limit, and then operated the program for another 4 years and in doing so had violated a limit and chosen to abort the simulation, the program would return the user to the 3 years of calculations point in the program.

(2) Ignoring Limits. As another option, the user can choose to ignore the limits and continue the simulation. If this option is chosen, the program resets the limit to the default value (0 for low limits and 9999 for high limits), warns the user that the limit has been reset, and then continues with the calculations.

(3) Back Up One Quarter. The user may also choose to back up one quarter in the simulation. The purpose of this option, is to allow the user to go back to

the point in the calculations, just before the limit was violated, allowing the user to then change some of the parameters, and then continue with the calculations. After completing calculations for one fewer quarter, the program automatically returns the user to the Selection Menu of Figure 3.4.

To continue with the above example, pressing "1" will cause the program to ignore the limit and continue the simulation. The program automatically sends the user a message that it has reset the high limit of Fourth Tour Professional Training to 9999, and that it is continuing calculations.

The program will next show the screen presented by Figure 3.27.

After 3 year(s) and 1 quarter(s),
The constraint for Third Tour Prof Trng has been exceeded.
There are 360 officers there now.
The high limit is 350
What do you desire to do now?

0. Abort simulation. Return to selection menu.
1. Ignore limits and continue simulation.
2. Back up one quarter in the simulation.

Figure 3.27 Second Constraint Violation Screen

This is the result of the user having earlier created a high limit for Third Tour Professional Training of 350 officers. This limit is exceeded after 3 years and 1 quarter of calculations. This time pressing "2" to back up one quarter in the simulation, the program will commence calculations on the data as configured just prior to commencing the calculations during which the limit was violated. Then the program will do calculations for one fewer quarter, i.e. for 3 years and 0 quarters. The program sends the user a message to this effect.

After completing calculations, the program returns the user to the Selection Menu of Figure 3.4. Note that the top line of the screen now indicates that 3 years and 0 quarters calculations have been completed. The user has now the choice to display data in various forms, in order to determine why the high limit was exceeded in the Third Tour Professional Training. The user also has the option to change data, e.g. the transfer path percentages that send people to Third Tour Professional Training or simply change the high limit for Third Tour Professional Training. As an example, pressing "7" to change data the change data menu will now appear on the screen (Figure 3.15). Pressing "4" to change high limit data, the Change High Limits Selection Screen (Figure 3.21) will be displayed. Pressing "3<CR>" to select THIRD TOUR and "A" to select PROFESSIONAL TRAINING, the Change High Limit Screen (Figure 3.22) is now displayed for Third Tour Professional Training. Entering "9999<CR>" to change the high limit from 350 to 9999and pressing "N" at the prompt asking if the user wants to make another change, the program finally returns to the Selection Menu of Figure 3.4.

b. Resetting Year and Quarter Count

To continue with another example, pressing "9" to commence calculations, the program will first ask if the user desires to reinitialize (set to zero) the years and quarters. So far the user has conducted calculations for 3 years and 0 quarters. If the user wants to maintain the year and quarter counters at 3 years and 2 quarters the user should answer no, by pressing "N", and at the prompt asking for how many years and quarters to run the model, he should enter "0<CR>" for 0 years and "2<CR>" for 2 quarters. The model will then perform calculations for 2 quarters and return to the Selection Menu of Figure 3.4. Now the top line reads that 3 years and 2 quarters calculations have

been completed. If the user responds yes, to the question 'Do you desire to reinitialize (set to zero) yrs & qtrs? Y/N' the program will reset the year counter and the quarter counter to zero. This will not affect the status of the assignment data, that still has 3 years and 0 quarters calculations completed. If the user now chooses to conduct 0 years and 2 quarters calculations, after the program has completed the calculations and returns to the selection menu, the top line would read that "0 years and 2 quarters calculations had been completed." Actually, this calculation was performed on top of 3 years and 0 quarters of previous calculations.

The user now could look over the data by choosing one or more of the several displays available. The user could also change some of the data, and continue calculations, or reinitialize some of the data and possibly start over again other wise the user may press "0" to indicate that he is finished.

8. Saving Data to a Disk File

The program then asks if the user desires to save any of the data to a disk file. This option will allow the user to choose to save some or all of the data that he has been using to a disk file named by the user. For instance, possibly the user wants to quit for the time being and resume his analysis later. In the above example there is assignment data as it exists after 3 years and 2 quarters of calculations that the user may wish to save. He has also changed high limits in two cases, and may want to save that data. Pressing "Y" will indicate that the user wants to save data to a disk file. Figure 3.28 is the next screen the user will see.

```
This procedure provides you with the opportunity to
save the data as presently configured in the program
to a disk file named by you.

                Press any key to continue . . .
```

Figure 3.28 Save Data to Disk File Caption Screen

Pressing any key, Figure 3.29 is the next screen the user will see.

```
Which data do you wish to save?

Data:                                              Data files:

0.  None/finished saving data.
1.  Number of officers at each assignment:         Nodes
2.  Transfer path percentages:                     Arcs
3.  Assignment tour lengths:                       Length
4.  High limits:                                   Hilimit
5.  Low limits:                                    Lolimit

Type your selection [0,1,2,3,4,5]:
```

Figure 3.29 Save Data Selection Menu Screen

The program provides the user with the opportunity to save any of the data as currently configured in the program. The data files listed are those that are currently in use. The data in these files has not been changed. The current configuration of the data is stored in the program's memory. Pressing "4" to save the high limit data to a disk file, Figure 3.30 is the next screen the user will see.

```
Enter output filename (maximum of 8 letter)
UNLESS YOU DESIRE TO OVERWRITE PREVIOUS DATA, DO NOT USE
HILIMIT
```

Figure 3.30 Save Data Filename Screen

The program prompts the user to enter a filename. Caution: The user must enter a filename, at least one letter

long and then press the carriage return. Just pressing the carriage return may cause the program to operate incorrectly. The following characters may not be used in the filename:

? . , ; : = * / \ + " < >

The following are examples of illegal filenames:

BAD/ONES - Illegal character (do not use "/")
WRONG.ONE - Illegal character (do not use ".")
TOOOOLONG - Too many characters

The following are examples of legal filenames:

NEWHIGH HIGHDATA USERHIGH GOODNAME

The user is warned about using the filename HILIMIT. If the user chooses that name, the default data stored in the file HILIMIT, will be destroyed permanently. As an example, the user may type the filename "HIGHTEST<CR>". It is not necessary to type the filename in uppercase. The program, and the operating system accepts either case. The program then dumps the currently configured data from program memory, to the disk file named HIGHTEST, sends a message to the user to let him know that this is happening (see bottom of screen), and then returns the user to the Save Data Selection Menu (Figure 3.29). the user could also save the assignment data by pressing 1 at this point.

9. Replacing Default Data Files

Pressing "0" to indicate finished saving data, Figure 3.31 will be the next screen the user will see.

The program now provides the user with the option to replace the default data files, (the files that

Now that you have saved that data to your own file, do you desire to use it to replace the base data? (Y/N)

Figure 3.31 First Replace Data Screen

automatically load in if the user makes no changes to the input data files at the first screen) with the currently configured data in the program's memory. As the next screen warns, this step is a very serious one. It is a one way change, and once done, the previous default data is lost forever. The user is probably better advised to not replace the base data file's data in most instances. He may still use the file just created if at the first screen in the program when asked whether the user desires to change the input data files he answers in the affirmative. However, if the for this example, the user answers yes by pressing "Y" to the query in Figure 3.31, the program will show Figure 3.32 on the next screen.

You have chosen to save the data as it is currently calculated by the program, to be the new initialization data.

This is a serious step.

You will be losing the default data established in the model.

Press any key to continue . . .

Figure 3.32 Second Replace Data Screen : WARNING

Pressing any key, Figure 3.33 is the next screen the user will see.

The program here provides the user with a menu to choose which data file the user wants to replace permanently. The user is again cautioned that this is a serious step that cannot be undone. If the user selected 1 through 5, the program will give the user one more

```
Which data do you desire to change PERMANENTLY?
Data:                                                Data files:

0.  None/finished replacing data.
1.  Number of officers at each assignment:           Nodes
2.  Transfer path percentages:                       Arcs
3.  Assignment tour lengths:                         Length
4.  High limits:                                     Hilimit
5.  Low limits:                                      Lolimit

Type your selection [0,1,2,3,4,5]:
```

Figure 3.33 Replace Data PERMANENTLY Selection Menu Screen

opportunity to change his mind, and then replace the data as selected by the user. The program then returns to the Replace Data PERMANENTLY Selection Menu Screen of Figure 3.33 to give the user the opportunity to replace some other data. Pressing "0" will cause the program to leave this section of the program without making any replacements.

Current use of the program is now completed and the user is shown an ending logo on the screen.

IV. MAINTENANCE OF THE MODEL AND FUTURE DEVELOPMENT

A. MAINTAINING AND UPDATING THE DATABASE

As stated in Chapter III, there are six data files contained on Distribution Disk I.

NODES.NOD
NODEZERO.NOD
ARCS.ARD
LENGTH.LED
HILIMIT.HID
LOLIMIT.LID

These six files contain the default data the program will use to initialize the variables, unless the user changes the input data file. Because data relevant for the model was not directly available at the writing of this report, all data in the default files was contructed somewhat artificially, although care was taken to make such data reasonable realistic. It is hoped that this data will soon be replaced by actual data by the model's first user. The assignment data that consists of the number of officers at each assignment provided in file "NODES" was abstracted/excerpted from an inventory of surface warfare officers for 1983, generated by the SWOTOURS model [Ref. 7]. Although the SWOTOURS model does not indicate the number of officers in the same format as this model, combining different outputs of the SWOTOURS model, and crossing years of commissioned service (YCS) with tour numbers, enabled the author to fill the model with "realistic" data. The default transfer path percentages were generated by analyzing data from the SWOTOURS model and then estimating the transfer path percentages from each assignment in the network. The default assignment tour length data was generated by evaluating the surface warfare

officer career path depicted in the Unrestricted Line Officer Career Planning Guidebook [Ref. 1] and equating the length of tours described with the networks assignment tour lengths. The high and low limit default data files were arbitrarily set at 9999 and 0 respectively.

As discussed previously, the user can decide which file the program will use to initialize the variables, by choosing a file previously saved. The user can also change the number of accessions, the individual values of the transfer path percentage variables, the assignment tour length variables, and the high and low limit variables. After making such changes the user is provided an opportunity to save those changes to a file of his own choosing. The user is then provided the opportunity to replace the default data files with these new data files if he so chooses. When actual data becomes available, the program itself, can be used to update the above mentioned default data in just that way.

The only data the user cannot change directly during operation of the program, is the number of officers at each assignment. The network consists of eight activities by twelve tours resulting in ninety six assignments/nodes. Each assignment can be between 1 and 16 quarters long, thus requiring one thousand five hundred and thirty six data entries to fill the network. Since the data files are not encrypted for the convenience of the user, each file could be edited using any word processing software package. Caution: This is NOT recommended. Hidden editing/formatting commands may be unintentionally inserted into the data files, or the user may accidentally violate the format of the data, rendering the data file useless. A separate program has been included on Distribution Disk II called UPDATE.COM to be used exclusively to update the number of officers at each assignment if the user wishes to do that. This file cannot be used to update any of the other data

files, since it only formats data for use as "NODES" data. It is a somewhat long and tedious process to update the data using this UPDATE program, but it is not likely that it will be required to be used very often.

To use the UPDATE program, the user should simply place the "working copy" of Distribution Disk II into the drive, type "UPDATE<CR>" and follow the prompts. This program will create a new file called "NODES.NOD" on the "working copy" of Distribution Disk II. In order to use it with the program the user must copy the new "NODES.NOD" file to Distribution Disk I using whatever procedures his operating system requires. One example for accomplishing this task is placing the "working copy" of Distribution Disk I in drive A, and the "working copy" of Distribution Disk II in drive B, and typing "A:<CR>". Then at the prompt : A>, the user should type "COPY B:NODES.NOD A:NODES.NOD". The data files are not "operating system" dependent, and the data files can be used interchangeably between the IBM and ZENITH versions of the program.

B. REVIEWING USER DATA FILES

The user is reminded that the program takes up approximately 58 Kbytes of space on Distribution Disk I and the data files add another 62 Kbytes, so 120 Kbytes of space on the disk are required simply to operate the basic program. As the user operates the program again, and again, it is expected that data files will be created, to enable the user to go back later and evaluate the results or recreate the results. The distribution disk is capable of holding 360 Kbytes of files, provided the user does not also have the operating system on the working copy of the distribution disk, in which there is approximately 350 Kbytes available. Subtracting the 120 Kbytes used by the program leaves approximately 240 (or 230) Kbytes for storage of the user created (saved) files. The user is cautioned to

occasionally use his operating system capabilities to review the amount of files storage area available, to preclude attempting to save a data file using the program, and failing to do so due to insufficient disk space available.

C. REVISING THE MODEL

This program was written and compiled, using Turbo Pascal Version 3.0. The ".PAS" files will not run using earlier versions of Turbo Pascal. This does not impact on the user since the ".PAS" files are not used in the normal operation of the program. The listing contained in Appendix A, contain many comments to aid future programmers in determining "how an why" certain procedures perform their tasks within the program. The program cannot be expanded beyond eight rows and twelve columns without a major rewrite, but with some changes to the activities labels, this network flow model could be modified to simulate the career paths of any officer community in the Navy. User's unfamiliar with programming in Pascal and who do not have Turbo Pascal Version 3.0 or later to compile their changes, are advised against altering any of the programs. The user is reminded that simply making changes to the ".PAS" listing will have no impact on how the program runs, until the program has been recompiled. If the user does attempt modifications, please ensure that any changes are fully documented by comments, to allow future users to review modifications made to the program. The user is reminded that the programs on the master distribution disks should never be modified. They are shipped with write protect tabs installed, and these tabs should never be removed.

The program has been operationally tested, and several undocumented features (a.k.a. bugs) have been corrected, doubtless there are more which have not yet been discovered.

APPENDIX A

THE MAIN PROGRAM IS CONTAINED IN THE DISK FILE "SWOPATH.PAS"

```
(*****************************************************************************
*****************************************************************************)

PROGRAM SWOPATH;

{ This is the main program.  Here all global variables are defined,
  all procedures are forward referenced, and all include files are
  called.  The remainder of the parts that make up the model are
  subroutines, called procedures, which the main program calls as
  necessary to run the model. }


TYPE

   One_Dim_Real = ARRAY [1..12] OF REAL;
   Two_Dim_Int = ARRAY [1..12,'A'..'H'] OF INTEGER;
   Two_Dim_Real = ARRAY [1..12,'A'..'H'] OF REAL;
   Tre_Dim_RealCH = ARRAY [1..12,'A'..'H','A'..'H'] OF REAL;
   Tre_Dim_RealNR = ARRAY [1..12,'A'..'H',1..16] OF REAL;
   AString = STRING [80];
   STR_25 = STRING [25];
   One_Dim_StrNR = ARRAY [1..16] OF STR_25;
   One_Dim_StrCH = ARRAY ['A'..'H'] OF STR_25;

VAR

   Tour_sum: One_Dim_Real;
   Tour,                          { label for which tour }
   TLength: One_Dim_StrNR;        { label for length in quarters)
   Activity: One_Dim_StrCH;       { label for activity }
   HiLimit,                       { max number of officers by billet node }
   LoLimit,                       { min number of officers by billet node }
   Temp_Billet_Total,             { temporary number at each billet node }
   Billet_Total: Two_Dim_Real;    { sum of officers at each billet node }
   New_Length,                    { new length of tour (to be changed to ) }
   Old_Length,                    { old billet length (for record purposes) }
   Billet_Length: Two_Dim_Int;    { actual billet length }
   Transfer_Path: Tre_Dim_RealCH; { percent of officers to be transferred }
   Path_total,                    { percent totals  of transfer paths }
   Temp_Billet_Node,              { temporary number of officers at each node }
   Billet_Node: Tre_Dim_RealNR;   { number of officers at node by quarters left }
   Accessions,                    { number of officer accessions }
   Tempcount,
   NumberSeconds,
   I,
   Y,
   T,                             { tour number from 1 to 12 }
   L,                             { quarters left from 1 to 16 }
   Total,
   NRQuarters,                    { how many quarters model is to run }
   NRYears,                       { value established by user }
   Quarter,                       { counts from 1 to 4 }
   QtrCount,                      { variable keeps track of # of iterations }
   YrCount: INTEGER;              { variable keeps track of nr of years }
   K,                             { used for activity }
   A,                             { used for activity }
   Answer,
   Choice,
   Reinit,                        { used in years to indicate reinitialize }
```

```
    Display,                    { answer to choice of display }
    Skip,
    First,                      { Used to input info to Correct_choice }
    Last: CHAR;                 { Used to input info to Correct_choice }
    Stop_calc_A,
    Stop_calc_T,                { limit warning boolean }
    Final_totals,
    Violation,                  { if limits violated }
    Toomany,                    { hi limit }
    Toofew,                     { lo limit }
    Change,                     { desire to change }
    Initial,                    { initialize values }
    All_finished,               { completely finished with program }
    Finished,                   { finished with particular section }
    Ready,
    Replace,                    { activates replacent of initial data }
    Review,                     { look over display selections }
    Ok_answer,                  { Used in correct_answer procedure }
    Ok_choice: BOOLEAN;         { Used in correct_choice procedure }
    File_var,
    Output_name,                {                           }
    Nodedata,                   {  these are all            }
    Arcdata,                    {   used in data file       }
    Lengdata,                   {     creation and transfer }
    Hilidata,
    Lolidata: Str_25;           { Data file variable names }
    Choice_zero,
    Chg_var : AString;
    Temporary:REAL;
    Data_file: TEXT;

(.pa){ forward references for all procedures }

PROCEDURE Logo; FORWARD;

PROCEDURE Dirlist; FORWARD;
PROCEDURE Initialize ; FORWARD;
PROCEDURE Initialize_nodes ; FORWARD;
PROCEDURE Initialize_arcs ; FORWARD;
PROCEDURE Initlallze_lengths;  FORWARD;
PROCEDURE Initialize_hilimits; FORWARD;
PROCEDURE Initialize_lolimits; FORWARD;
PROCEDURE Initialize_labels; FORWARD;

PROCEDURE Selection_menu; FORWARD;
PROCEDURE Review_selections ; FORWARD;
PROCEDURE Calculations ; FORWARD;
PROCEDURE Ask_for_years ; FORWARD;

PROCEDURE All_billet_totals; FORWARD;
PROCEDURE Billet_totals (VAR T: INTEGER;
                         VAR A: CHAR    ); FORWARD;
PROCEDURE High_warning; FORWARD;
PROCEDURE Low_warning; FORWARD;

PROCEDURE Changes; FORWARD;
PROCEDURE Change_accessions; FORWARD;
PROCEDURE Change_arcs; FORWARD;
PROCEDURE Change_length; FORWARD;
PROCEDURE Change_hilimits; FORWARD;
PROCEDURE Change_lolimits; FORWARD;

PROCEDURE Dchoices; FORWARD;
PROCEDURE Dchanges; FORWARD;
PROCEDURE From_pathscrn; FORWARD;
PROCEDURE To_pathscrn; FORWARD;

PROCEDURE Disp_assign; FORWARD;
```

```
PROCEDURE Disp_tours; FORWARD;
PROCEDURE Disp_paths_from; FORWARD;
PROCEDURE Disp_paths_to; FORWARD;
PROCEDURE Disp_billets; FORWARD;

PROCEDURE Node_dump; FORWARD;
PROCEDURE Arc_dump; FORWARD;
PROCEDURE Length_dump; FORWARD;
PROCEDURE Hilln_dump; FORWARD;
PROCEDURE Lolin_dump; FORWARD;

PROCEDURE Invalid_answer; FORWARD;
PROCEDURE Wrong_answer; FORWARD;
PROCEDURE Correct_choice (VAR Choice, First, Last: CHAR;
                          VAR Ok_choice: BOOLEAN); FORWARD;
PROCEDURE Correct_answer (VAR Answer: CHAR;
                          VAR Ok_answer: BOOLEAN); FORWARD;
PROCEDURE Another_change; FORWARD;
PROCEDURE Strip (VAR File_var: STR_25); FORWARD;
PROCEDURE Blankline (X, Y: INTEGER); FORWARD;

PROCEDURE Save_data ; FORWARD;
PROCEDURE Replace_data; FORWARD;

{.pa} { link up all include files }

   {$I zenUTILS }                {% This reads "ibmUTILS" %}
                                 {% in IBM PC version     %}
   {$I Logo}

   {$I Initdata}

   {$I Selectio}

   {$I Calculat}

   {$I Totals}

   {$I Chgdata}

   {$I Screens}

   {$I Displays}

   {$I Datadump}

   {$I Answers}

   {$I Stordata}


{.pa}
{  The beginning of the MAIN Program  }

BEGIN

  Accessions := 350;
  Tempcount := 0;
  NRYears := 0;
  NRQuarters := 0;
  Quarter := 0;
  YRcount := 0;
  Review := false;    { to skip sample selections the first time }
  Initial := true; { to initialize the data the first time }
  New_Length [ T, A ] := Billet_Length [ T, A ];
  Logo;

    IF Initial THEN
```

```
    BEGIN ( initial if )
    Initialize;
    Change_accessions;
    Initial := false;  ( to prevent program from reinitializing)
    END; ( initialize if)

    REPEAT    ( repeat loop until allfinished )
    All_finished := false;  ( to ensure that we are not all finished )
    Ready :=  false;    ( establish that we are not ready for calcul )
    Change := false;    ( establish that initially we do not change )

      IF NOT All_finished THEN Selection_menu;

    UNTIL All_finished = true;

  Save_data;
  Clearscreen;
  GotoRC ( 12, 13);
  Color ( white, magenta);
  WRITELN ('              Have a nice day!              ');
  Color ( yellow, blue);

END.  ( SWOPATH main program )
```

APPENDIX B

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "ZENUTILS.PAS"
{**************************************************************************
**************************************************************************}

 {******************************************************
 **                                                  **
 **           MISCELLANEOUS UTILITIES                **
 **              H/Z-100 Version                     **
 **                                                  **
 ******************************************************}

{ This file contains the following procedures
  specifically written to be Heath/Zenith
  compatible

  ClearScreen
  Color
  GotoRC
  Center
  Tab
  GetOneChar
  Pause
  Strip
  Blankline

}


type
     ScreenColor = (Black,Blue,Red,Magenta,Green,
                    Cyan,Yellow,White);

procedure ClearScreen;

  {  This procedure simply clears the screen  }

begin

  ClrScr;

end;  {ClearScreen}

{**************************************************************************
**************************************************************************}


procedure Color(Foreground,Background : ScreenColor);
```

```
   {This procedure sets the Screen Colors according to the
    requested colors sent.  Colors are: Black,Blue,Green
    Cyan,Red,Magenta,Yellow,White }

begin

  write(chr(27),'m',chr(ord(Foreground)+48),
              chr(ord(Background)+48));

end; {Color}

{*****************************************************************************
*****************************************************************************}



procedure GotoRC(Row, Column : integer);

   {This provides direct cursor addressing. There is a built-
    in function GotoXY which asks for XY (or Col-Row) order. }

begin

  if Row > 24 then Row := 24;
  if Column > 80 then Column := 80;
  GotoXY(Column,Row);

end; {GotoRC}


{*****************************************************************************
*****************************************************************************}


procedure Center(Line : integer; Message : AString);

   {This procedure will center the Message in 80 columns
    on the screen.  Must pass message and line it's on}

var

 Count : integer;

begin

  if Length(Message) >79 then
  begin
  GotoRC(Line,1);
  writeln(Message);
  end

  else
  begin
  GotoRC(Line,((80-Length(Message)) div 2));
  write(Message);
  end; {begin-else}
  {end if-then-else}

end; {Center}

{*****************************************************************************
*****************************************************************************}


procedure GetOneChar(var UserAnswer : char);

   {This procedure gets exactly one character, converts to
    Upper case (if needed) and displays it on screen     }
```

```
begin

  read(Kbd,UserAnswer);
  UserAnswer := UpCase(UserAnswer);

end; {GetOneChar}

{##########################################################################
##########################################################################}


procedure Pause;

   { This routine merely waits for any key; if want to check
     for Control C for exit, use PauseQ below }

var

  AnyKey : char;

begin

  writeln;
  GotoRC(23,23);
  write('Press any key to continue . . .  ');
  GetOneChar(AnyKey);

end; { Pause }

{##########################################################################
##########################################################################}


PROCEDURE Strip ;

  { this procedure strips the filetype from a filename }

TYPE

   AString = STRING [80];

VAR

  A,
  E: ASTRING;
  X,
  Y,
  Z: INTEGER;

BEGIN

  X := LENGTH (File_var);
  Y := X-3;
  DELETE (File_var,Y,4);

END;

{##########################################################################
##########################################################################}


PROCEDURE Blankline;

{ This procedure blakns out a line given row and column number }
```

```
BEGIN

  GotoRC (X, Y);
  WRITE
('                                                          ');
  GotoRC (X,Y);

END; { Blankline }
```

APPENDIX C

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "IBMUTILS.PAS"
{##############################################################################
##############################################################################}

{##############################################
 ##                                          ##
 ##          MISCELLANEOUS UTILITIES         ##
 ##              IBM VERSION                 ##
 ##                                          ##
 ##############################################}

{ This file contains the following procedures
  specifically written to be IBM compatible

   ClearScreen
   Color
   GotoRC
   Center
   Tab
   GetOneChar
   Pause
   Strip
   Blankline

}

const

  HTab  : Char = ^I;
  Bell  : Char = ^G;
  LFeed : Char = ^J;
  Return: Char = ^M;

{##############################################################################
##############################################################################}


procedure ClearScreen;

   {  This procedure simply clears the screen  }

begin

  ClrScr

end;  {ClearScreen}

{##############################################################################
##############################################################################}


procedure Color(Foreground,Background : Integer);

   {This procedure sets the Screen Colors according to the
    requested colors sent.  Colors are: Black,Blue,Green
    Cyan,Red,Magenta,Yellow,White,Gray,Light Blue,Light Green,
    Light Cyan,Light Red,Yellow,High White }

begin

  Textbackground(Background);
```

```
  Textcolor(Foreground);

end; {Color}

{*************************************************************************
**************************************************************************}


procedure GotoRC(Row, Column : integer);

   {This provides direct cursor addressing. There is a built-
    in function GotoXY which asks for XY (or Col-Row) order. }

begin

  if Row > 24 then Row := 24;

  if Column > 80 then Column := 80;

  GotoXY(Column,Row);

end; {GotoRC}

{*************************************************************************
**************************************************************************}


procedure Center(Line : integer; Message : AString);

   {This procedure will center the Message in 80 columns
    on the screen.  Must pass message and line it's on}

var

  Count : integer;

begin

    if Length(Message) >79 then
    begin
    GotoRC(Line,1);
    writeln(Message);
    end

    else
    begin
    GotoRC(Line,((82-Length(Message)) div 2));
    write(Message);
    end; {begin-else}

    {end if-then-else}
end; {Center}

{*************************************************************************
**************************************************************************}


procedure Tab(NumberSpaces:integer);

{ This procedure moves the cursor over the appropriate
 number of spaces requested }

var

  Count : integer;

begin
```

```
  for Count := 1 to NumberSpaces do
  begin
  write(' ')
  end;  (begin)

(end for)
end;  (begin)

{*************************************************************************
*************************************************************************}


procedure GetOneChar(var UserAnswer : char);

(This procedure gets exactly one character, converts to
 Upper case (if needed) and displays it on screen     )

begin
      read(Kbd,UserAnswer);
      UserAnswer := UpCase(UserAnswer);
end; (GetOneChar)


{*************************************************************************
*************************************************************************}


procedure Pause;

( This routine merely waits for any key; if want to check
    for Control C for exit, use Paused below )

var
  AnyKey : char;

begin

  writeln;
  Tab(23);
  write('Press any key to continue . . . ');
  GetOneChar(AnyKey);

end; ( Pause )

{*************************************************************************
*************************************************************************}


PROCEDURE Strip ;

( This procedure removes the filetype from a filename )

TYPE

  AString = STRING [80];

VAR

  A,
  E: ASTRING;
  X,
  Y,
  Z: INTEGER;

BEGIN

  X := LENGTH (File_var);
  Y := X-3;
```

```
  DELETE (File_var,Y,4);
END;

{##########################################################################
##########################################################################}

PROCEDURE Blankline;
 { This procedure blanks out an entire line.  Must pass row and column }
BEGIN
  GotoRC (X, Y);
  WRITE
('                                                                      ');
  GotoRC(X, Y);
END; { Blankline }
```

APPENDIX D

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "LOGO.PAS"
{##########################################################################
##########################################################################}

{  This file contains the procedure

    Logo

}

PROCEDURE Logo;

    { This procedure calls up the initial screen logo.}

BEGIN

  ClearScreen;
  Color (white,blue);
  ClearScreen;
  Color(white, magenta);
  Center(10,'Welcome to SWOPATH');
  Center(12,'The Surface Warfare Officer');
  Center(14,'Career Path Model');
  Center (18,'Version 1.0');              { Be sure to change version number }
  Center (20,'27 September 1985');
  Color(white, blue);                    { and date if modifications are made }
  GotoRC (23, 1);
  DELAY (1500);

END; {Logo}
```

APPENDIX E

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "INITDATA.PAS"
(*****************************************************************
*****************************************************************)

{ This file contains the following procedures

    Initialize
    Initialize_nodes
    Initialize_arcs
    Initialize_lengths
    Initialize_labels
    Initialize_hilimits
    Initialize_lolimits
    DirList
}


PROCEDURE Initialize;

{ This procedure provides the displays and mechanics to call up
  the data initialization procedures }

VAR

  Ok: BOOLEAN;
  I,Y : INTEGER;
  Default : Str_25;

BEGIN

  Finished := false;

    IF Initial THEN
    BEGIN
    Nodedata := 'Nodes';
    Arcdata := 'Arcs';
    Lengdata := 'Length';
    Hilidata := 'Hilimit';
    Lolidata := 'Lolimit';

      REPEAT { until finished }
      Chg_var :=
      'Do you want to change the input data files from those listed below?';
      Choice_zero :='No changes wanted/finished changes.';
      Bchanges;

        IF Choice = '0' THEN Finished := true

        ELSE
        BEGIN
        Clearscreen;
        GotoRC (5, 1);
        WRITELN ('  Remember, you must choose a data file that you have ');
        WRITELN ('      previously saved.  Those are listed below.');
        WRITELN;
        Color (red, white );
        Dirlist;
        Color (white, blue);
        WRITELN;WRITELN;
        GotoRC (11, 15);
        WRITELN ('  Enter the input filename');
```

```
    Color (blue, white);
    GotoRC (13, 15);
    WRITELN (' typed exactly as listed above,');
    Color (white, blue);
    GotoRC (15, 15);
    WRITELN (' that you choose to use: ');

      REPEAT   ( Ok true )

        IF Choice = '1' THEN Default := 'Nodes';
        IF Choice = '2' THEN Default := 'Arcs';
        IF Choice = '3' THEN Default := 'Length';
        IF Choice = '4' THEN Default := 'Hilimit';
        IF Choice = '5' THEN Default := 'Lolimit';

      GotoRC (20,15);
      WRITE ('Type ', Default,'  if you want to exit back to menu.');
      Blankline (15, 40);
      GotoRC (15, 40);
      READ (File_var);

        IF Choice = '1' THEN File_var := CONCAT (File_var,'.nod');
        IF Choice = '2' THEN File_var := CONCAT (File_var,'.ard');
        IF Choice = '3' THEN File_var := CONCAT (File_var,'.led');
        IF Choice = '4' THEN File_var := CONCAT (File_var,'.hid');
        IF Choice = '5' THEN File_var := CONCAT (File_var,'.lod');

      ASSIGN (Data_file, File_var);(This procedure determines if there )
      ($I-) RESET (Data_file) ($I+);(is such a file as selected by user.)
      Ok := (IOresult = 0);
      Strip (File_var);

        IF NOT Ok THEN
        BEGIN
        Blankline (24, 1);
        Color (white, red);
        WRITE (' Cannot find file ', File_var,' please try again.');
        DELAY (1500);
        Color (white, blue);
        Blankline (24, 1);
        END; ( Ok IO result if )

      UNTIL Ok = True;

      IF Choice = '1' THEN Nodedata := File_var;
      IF Choice = '2' THEN Arcdata  := File_var;
      IF Choice = '3' THEN Lengdata := File_var;
      IF Choice = '4' THEN Hilidata := File_var;
      IF Choice = '5' THEN Lolidata := File_var;

    END; ( if choice = 0 )

  UNTIL Finished = true;

Initialize_labels;
Initialize_nodes;
Initialize_arcs;
Initialize_lengths;
Initialize_hillmits;
Initialize_lolimits;
Final_totals := true;
All_billet_totals;
Blankline (24,1);
END; (if Initial )

IF NOT Initial THEN
BEGIN
```

```
      REPEAT ( Until finished true )
      Chg_var := 'Which data do you desire to reinitialize?';
      Choice_zero := 'None/finished reinitializing data.';
      Dchanges;

        IF NOT Ok_choice THEN Initialize;

        IF Choice = '0' THEN Finished := true;

        IF Choice = '1' THEN
        BEGIN
        Initialize_nodes;
        Arcdata := CONCAT (Arcdata,'.ard');
        Lengdata := CONCAT (Lengdata,'.led');
        Hilidata := CONCAT (Hilidata,'.hid');
        Lolidata := CONCAT (Lolidata,'.lod');
        END;  ( If choice 1 )

        IF Choice = '2' THEN
        BEGIN
        Initialize_arcs;
        Nodedata := CONCAT (Nodedata,'.nod');
        Lengdata := CONCAT (Lengdata,'.led');
        Hilidata := CONCAT (Hilidata,'.hid');
        Lolidata := CONCAT (Lolidata,'.lod');
        END;  ( If choice 2 )

        IF Choice = '3' THEN
        BEGIN
        Initialize_lengths;
        Nodedata := CONCAT (Nodedata,'.nod');
        Arcdata := CONCAT (Arcdata,'.ard');
        Hilidata := CONCAT (Hilidata,'.hid');
        Lolidata := CONCAT (Lolidata,'.lod');
        END;  ( If choice 3 )

        IF Choice = '4' THEN
        BEGIN
        Initialize_hilimits;
        Nodedata := CONCAT (Nodedata,'.nod');
        Arcdata := CONCAT (Arcdata,'.ard');
        Lengdata := CONCAT (Lengdata,'.led');
        Lolidata := CONCAT (Lolidata,'.lod');
        END; (If choice 4 )

        IF Choice = '5' THEN
        BEGIN
        Initialize_lolimits;
        Nodedata := CONCAT (Nodedata,'.nod');
        Arcdata := CONCAT (Arcdata,'.ard');
        Lengdata := CONCAT (Lengdata,'.led');
        Hilidata := CONCAT (Hilidata,'.hid');
        END; ( if choice 5 )

      UNTIL Finished = true;

    Final_totals := true;
    All_billet_totals;
    Clearscreen;

    END; ( if NOT initial )

END; ( Initialize procedure)


{*****************************************************************************
*****************************************************************************}
```

```
{.pa}

PROCEDURE Initialize_nodes;

 { This procedure reads assignment data from
   a data file on the disk }

BEGIN

  Blankline (23, 10);
  GotoRC (23, 1);
  WRITE ('  Initializing the number of officers at nodes ',
              '', 'using the file: ', Nodedata);
  Nodedata := CONCAT (Nodedata,'.nod');
  ASSIGN (Data_file,Nodedata);
  RESET (Data_file);

    WHILE NOT EOF(Data_file) DO
    BEGIN
    READ (Data_file, T );

      IF NOT EOF(Data_file) THEN
      BEGIN
      READ (Data_file, Skip);
      READ (Data_file, A );
      READ (Data_file, Skip);
      READ (Data_file, L );
      READ (Data_file, Skip);
      READLN (Data_file, Billet_node [ T, A , L ]);
      END;  { if not end of file }

    END; { end while not end of file }

  CLOSE (Data_file);

END;  { Initialize nodes }


{*************************************************************************
*************************************************************************}

{.pa}

PROCEDURE Initialize_Arcs;

{ This procedure reads transfer path percentages from
  a data file on the disk }

VAR

  T,
  L: INTEGER;
  A,
  K: CHAR;
  Data_file: TEXT;

BEGIN

  Blankline (23, 10);
  GotoRC (23, 1);
  WRITE ('  Initializing the transfer path percentages ',
                    '','using the file: ',Arcdata);
  Skip := ' ';
  Arcdata := CONCAT (Arcdata, '.ard');
  ASSIGN (Data_file,Arcdata);
  RESET (Data_file);
```

```
    WHILE NOT EOF(Data_file) DO
    BEGIN
    READ (Data_file,T);

      IF NOT  EOF(Data_file) THEN
      BEGIN
      READ (Data_file, Skip);
      READ (Data_file, A);
      READ (Data_file, Skip);
      READ (Data_file, K);
      READLN (Data_file,Transfer_Path [ T, A, K ]);
      END;  ( end if not end of file )

    END;  ( end while not end of file )

  CLOSE (Data_file);

END;  ( Initialize arcs )

{##########################################################################
##########################################################################}


(.pa)

PROCEDURE Initialize_lengths;

( This procedure reads the assignment tour length data
  from a data file on the disk )

VAR

  T: INTEGER;
  A: CHAR;
  Data_file: TEXT;

BEGIN

  Blankline (23, 10);
  GotoRC (23, 1);
  WRITE (' Initializing assignment tour lengths using the file: ', Lengdata);
  Lengdata := CONCAT (Lengdata, '.led');
  ASSIGN (Data_file, Lengdata);
  RESET (Data_file);

    WHILE NOT EOF(Data_file) DO
    BEGIN
    READ (Data_file, T );

      IF NOT EOF(Data_file) THEN
      BEGIN
      READ (Data_file, Skip);
      READ (Data_file, A );
      READLN (Data_file,Billet_length [ T, A ]);
      END; {end if not end of file )

    END; ( end while not end of file )

  CLOSE (Data_file);

END;  ( initialize lengths )


{##########################################################################
##########################################################################}

(.pa)
```

```
PROCEDURE Initialize_labels;

( This procedures establishes the display labels for
  tours, activities, and quarters left )

BEGIN

  Tour [1] := 'First Tour ';
  Tour [2] := 'Second Tour ';
  Tour [3] := 'Third Tour ';
  Tour [4] := 'Fourth Tour ';
  Tour [5] := 'Fifth Tour ';
  Tour [6] := 'Sixth Tour ';
  Tour [7] := 'Seventh Tour ';
  Tour [8] := 'Eighth Tour ';
  Tour [9] := 'Ninth Tour ';
  Tour [10] := 'Tenth Tour ';
  Tour [11] := 'Eleventh Tour ';
  Tour [12] := 'Twelfth Tour ';
  Activity [ 'A' ] := 'Prof Trng ';
  Activity [ 'B' ] := 'Prof Educ ';
  Activity [ 'C' ] := 'WashDC ';
  Activity [ 'D' ] := 'Shore (CONUS) ';
  Activity [ 'E' ] := 'Fleet Unit ';
  Activity [ 'F' ] := 'Afloat Staff ';
  Activity [ 'G' ] := 'Shore (OUTUS) ';
  Activity [ 'H' ] := 'Separation ';
  TLength [ 1 ] := ' with  1 qtr  left';
  TLength [ 2 ] := ' with  2 qtrs left ';
  TLength [ 3 ] := ' with  3 qtrs left';
  TLength [ 4 ] := ' with  4 qtrs left';
  TLength [ 5 ] := ' with  5 qtrs left';
  TLength [ 6 ] := ' with  6 qtrs left';
  TLength [ 7 ] := ' with  7 qtrs left';
  TLength [ 8 ] := ' with  8 qtrs left';
  TLength [ 9 ] := ' with  9 qtrs left';
  TLength [ 10 ] := ' with 10 qtrs left';
  TLength [ 11 ] := ' with 11 qtrs left';
  TLength [ 12 ] := ' with 12 qtrs left';
  TLength [ 13 ] := ' with 13 qtrs left';
  TLength [ 14 ] := ' with 14 qtrs left';
  TLength [ 15 ] := ' with 15 qtrs left';
  TLength [ 16 ] := ' with 16 qtrs left';

END; { Initialize labels }


(*****************************************************************************
*****************************************************************************)

(.pa)


PROCEDURE Initialize_hilimits;

{ This procedure reads high limit data from a data file on the disk }

VAR

  I: INTEGER;
  A: CHAR;
  Data_file: TEXT;

BEGIN

  Blankline (23, 10);
  GotoRC (23, 1);
  WRITE ('  Initializing the high limits using the file: ', Hilidata) ;
```

```
  Hilidata := CONCAT (Hilidata,'.hid');
  ASSIGN (Data_file, Hilidata);
  RESET (Data_file);

    WHILE NOT EOF(Data_file) DO
    BEGIN
    READ (Data_file, T );

      IF NOT EOF(Data_file) THEN
      BEGIN
      READ (Data_file, Skip);
      READ (Data_file, A );
      READ (Data_file, Skip);
      READLN (Data_file, HiLimit [ T, A ]);
      END; { end if not end of file if  }

    END; { end while not EOF loop }

  CLOSE (Data_file);

END; { Hilimit procedure }

{##########################################################################
##########################################################################}

{.pa}

PROCEDURE Initialize_lolimits;

{ This procedure reads low limit data from a disk file }

VAR

  T: INTEGER;
  A: CHAR;
  Data_file: TEXT;

BEGIN

  Blankline (23, 10);
  GotoRC (23, 1);
  WRITE ('  Initializing Low Limits using the file: ', Lolidata);
  Lolidata := CONCAT (Lolidata, '.lod');
  ASSIGN (Data_file, Lolidata);
  RESET (Data_file);

    WHILE NOT EOF(Data_file) DO
    BEGIN
    READ (Data_file, T );

      IF NOT EOF(Data_file) THEN
      BEGIN
      READ (Data_file, Skip);
      READ (Data_file, A );
      READ (Data_file, Skip);
      READLN (Data_file, Lolimit [ T, A ]);
      END; { end if not end of file  }

    END; { end while not end of file  }

  CLOSE (Data_file);

END; { Lolimit procedure }


{##########################################################################
##########################################################################}
```

```
{.pa}


PROCEDURE DirList;
{
      This is a simple program to list out the directory of the
      current (logged) drive in accordance with filetype chosen.
       MODIFIED BY RBA  }

type

  Char12arr           = array [ 1..12 ] of Char;
  String20            = string[ 20 ];
  RegRec =
    record
      AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : Integer;
    end;

var

  Regs                : RegRec;
  DTA                 : array [ 1..43 ] of Byte;
  Mask                : Char12arr;
  NamR                : String20;
  I,Y,Error, I          : Integer;

begin { main body of program DirList }

  FillChar(DTA,SizeOf(DTA),0);        { Initialize the DTA buffer }
  FillChar(Mask,SizeOf(Mask),0);      { Initialize the mask }
  FillChar(NamR,SizeOf(NamR),0);      { Initialize the file name }
  Regs.AX := $1A00;                   { Function used to set the DTA }
  Regs.DS := Seg(DTA);                { store the parameter segment in DS }
  Regs.DX := Ofs(DTA);                {    "     "       "     offset in DX }
  MSDos(Regs);                        { Set DTA location }
  Error := 0;

    IF Choice = '1' THEN Mask := '????????.mod';
    IF Choice = '2' THEN Mask := '????????.ard';
    IF Choice = '3' THEN Mask := '????????.led';
    IF Choice = '4' THEN Mask := '????????.hld';
    IF Choice = '5' THEN Mask := '????????.lod';
        { Use global search }
  Regs.AX := $4E00;                   { Get first directory entry }
  Regs.DS := Seg(Mask);               { Point to the file Mask }
  Regs.DX := Ofs(Mask);
  Regs.CX := 22;                      { Store the option }
  MSDos(Regs);                        { Execute MSDos call }
  Error := Regs.AX and $FF;           { Get Error return }
  I := 1;                             { initialize 'I' to the first element }

  if (Error = 0) then

    repeat
    NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
    I := I + 1;
    until not (NamR[I-1] in [' '..'~']) or (I>20);

  NamR[0] := Chr(I-1);                { set string length because assigning }
                                      { by element does not set length }

    IF (Error = 0) THEN
    BEGIN
    I := LENGTH (NamR);
    Y := I-4;
    DELETE (NamR,Y,4);
    WRITELN(NamR);
```

```
    END;

    while (Error = 0) do begin
    Error := 0;
    Regs.AX := $4F00;           { Function used to get the next }
                                { directory entry }
    Regs.CX := 22;              { Set the file option }
    MSDos( Regs );              { Call MSDos }
    Error := Regs.AX and $FF;   { get the Error return }
    I := 1;

      repeat
      NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
      I := I + 1;
      until not (NamR[I-1] in [' '..'^'] ) or (I > 20);

    NamR[0] := Chr(I-1);

      IF (Error = 0) THEN
      BEGIN
      X := LENGTH (NamR);
      Y := X-4;
      DELETE (NamR,Y,4);
      WRITELN(NamR);
      END;  { error if}

    END; { while do loop }

end; { of procedure DirList  }
```

APPENDIX F

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "SELECTIO.PAS"
(##########################################################################
##########################################################################)

{  This file contains the following procedures

    Selection_menu
    Review_selection

}

PROCEDURE Selection_menu;

{ This procedure provides the selection menu display and
  choice evaluation mechanics }

VAR

  Choice: CHAR;
  Ok_choice: BOOLEAN;

BEGIN

  REPEAT { until choice is correct }
  ClearScreen;
  WRITELN;
  WRITELN ('  Calculations have been completed for ',YrCount,' years and  ',
             Quarter,'   quarter(s).');
  GotoRC (6, 1);
  WRITELN ('                Choose one of the following selections.');
  GotoRC (8, 1);
  WRITELN ('        0.  All finished.');
  WRITELN ('        1.  Review the display selections.');
  WRITELN ('        2.  Display an activity v.s. all tours');
  WRITELN ('        3.  Display a tour v.s. all activities');
  WRITELN ('        4.  Display the transfer paths FROM an assignment');
  WRITELN ('        5.  Display the transfer paths TO an assignment');
  WRITELN ('        6.  Display the assignments by quarters left');
  WRITELN ('        7.  Change data values.');
  WRITELN ('        8.  Reinitialize data values.');
  WRITELN ('        9.  Ready for calculations.');
  GotoRC (19, 5);
  WRITE ('Please type in the number of your selection [0,1,2,3,4,5,6,7,8,9]: ');
  READ (Kbd, Choice); {read input into var selection}
  First:= '0'; Last:= '9'; Ok_choice := true;   { inputs for correct_choice }
  Correct_choice (Choice, First, Last, Ok_choice);{       procedure.        }
  UNTIL Ok_choice = true;

  IF Choice = '0' THEN All_finished := true;
  IF Choice = '1' THEN Review_selections;
  IF Choice = '2' THEN Disp_assign;
  IF Choice = '3' THEN Disp_tours;;
  IF Choice = '4' THEN Disp_paths_from;
  IF Choice = '5' THEN Disp_paths_to;
  IF Choice = '6' THEN Disp_billets;
  IF Choice = '7' THEN Changes;

  IF Choice = '8' THEN
  BEGIN
  Initial := false;
  Initialize;
```

```
  END; { if choice = 8 }

  IF Choice = '9' THEN Ask_for_years;

END; (Selection_Menu)

{*************************************************************************
*************************************************************************}

{.pa}

PROCEDURE Review_selections;

 { This procedures contains the sample displays }

BEGIN

  ClearScreen;
  Center (04,'*  Activity v.s. all tours  *');
  Center (06,'Displays the number of officers');
  Center (07,'at a specific assignment for all tours.');
  GotoRC (9,10);
  WRITELN ('For example the display will look like:');
  WRITELN;
  WRITELN ('                      Prof Educ ');
  WRITELN;
  WRITELN ('                            TOURS..... ');
  WRITELN ('                       First     Second    Third ... ');
  WRITELN;
  WRITELN('                       400       325      643');
  Color ( black, white);
  Center (18,' Selection number 2');
  Color ( white, blue);
  GotoRC (23, 1);
  Pause;

  ClearScreen;
  Center (04,'*  Tour v.s. all activities  *');
  Center (06,'Displays the  number of officers');
  Center (07,'at a specific tour for all assignments. ');
  GotoRC (9, 10);
  WRITELN('For example the display will look like:');
  WRITELN;
  WRITELN('                              TOUR');
  WRITELN('             ACTIVITIES');
  WRITELN('                              First');
  WRITELN('              Prof Educ       400 ');
  WRITELN('              Pro Trng        342 ');
  WRITELN('              WashDC          981 ');
  Color ( black, white);
  Center (20,' Selection number 3');
  Color ( white, blue);
  GotoRC (23, 1);
  Pause;

  ClearScreen;
  Center (04,'*  Transfer paths FROM an assignment  *');
  Center (06,'Displays all of the transfer path percentages');
  Center (07,'and the number of officers transferred, from a specific assignment.');
  GotoRC (10,10);
  WRITELN ('For example the display will look like:');
  WRITELN;
  WRITELN ('                       to 3rd tour Pro ed');
  WRITELN ('                     /   31% or 5 officers');
  WRITELN ('                    / ');
  WRITELN ('  2nd tour Pro ed ---- to 3rd tour Pro trng ');
  WRITELN ('     9 officers     \    14% or 2 officers');
```

```
  WRITELN ('     transferred     \');
  WRITELN ('                       to 3rd tour WashDC ');
  WRITELN ('                         11% or 2 officers');
  Color ( black, white);
  Center (21,' Selection number 4');
  Color ( white, blue);
  GotoRC (23, 1);
  Pause;

  ClearScreen;
  Center (04,'"  Transfer paths TO an assignment  "');
  Center (06,'Displays the number of officers transferred');
  Center (07,'INTO an assignment from those assignments in');
  Center (08,'the previous tour.  ');
  GotoRC (10,10);
  WRITELN ('For example the display will look like:');
  WRITELN;
  WRITELN (' FROM 2nd tour Pro ed  -\                              ');
  WRITELN ('  31% or 5 officers      \                             ');
  WRITELN ('                          \        TO                 ');
  WRITELN (' FROM 2nd Tour Pro trng----\   3rd Tour Shore (CONUS)   ');
  WRITELN ('   14% or 2 officers       >                           ');
  WRITELN ('                          /     9 officers            ');
  WRITELN (' FROM 2nd Tour Wash DC ---/     transferred            ');
  WRITELN ('   11% or 2 officers     /                             ');
  Color ( black, white);
  Center (21,' Selection number 5');
  Color ( white, blue);
  GotoRC (23, 1);
  Pause;

  ClearScreen;
  Center (04,'"  Assignments by quarters left  "');
  Center (06,'Displays the number of officers at a specific ');
  Center (07,'assignment by how many quarters they have left there');
  GotoRC (10, 10);
  WRITELN ('For example the display will look like:');
  WRITELN;
  WRITELN ('     Officers assigned:      ');
  WRITELN ('');
  WRITELN ('        43           with 1 qtr  left');
  WRITELN ('         8           with 2 qtrs left');
  WRITELN ('         9           with 3 qtrs left');
  WRITELN ('');
  WRITELN ('');
  WRITELN ('');
  Color ( black, white);
  Center (21,' Selection number 6');
  Color (white, blue);
  GotoRC (23, 1);
  Pause;

END;  ( review )
```

APPENDIX G

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "CALCULAT.PAS"
(*************************************************************************
*************************************************************************)


{ This file contains the following procedures

    Ask_for_years
    Calculations

}


PROCEDURE Ask_for_years;

VAR

  Years_wanted,
  Qtrs_wanted: INTEGER;

BEGIN

  REPEAT { until NRYears answer is an integer }

    REPEAT { until NRQuarters answer is an integer }
    Blankline (23, 1);
    WRITE ('      Choose 0 years & 0 quarters to quit.');
    DELAY (2000);
    Blankline (23, 1);

      REPEAT  { until correct answer }
      Blankline (24, 1);
      WRITE ('  Do you desire to reinitialize','',
                          '(set to zero) yrs & qtrs? (Y/N) ');
      READ (Kbd, Answer);
      Correct_answer (Answer, Ok_answer);
      UNTIL Ok_answer = true;

    IF Answer in ['Y','y'] THEN
    BEGIN
    NRQuarters := 0;
    YRCount := 0;
    Quarter := 0;
    TempCount := 0;
    END; {reset yrs and qtrs to zero }

    Blankline (24, 1);
    WRITE ('How many years: <CR>and quarters: <CR>do you want to run the model?');
    GotoRC (24, 16);

    {$I-} READ (Years_wanted) {$I+};
    Ok_answer := (IORESULT = 0); {TURBO PASCAL function to check input type }

      IF NOT Ok_answer THEN Invalid_answer;

    UNTIL Ok_answer = true;

  GotoRC (24, 34);
  {$I-} READ (Qtrs_wanted) {$I+};
  Ok_answer := (IORESULT = 0); {TURBO PASCAL function to check input type }
```

```
    IF (Qtrs_wanted < 0) OR (Qtrs_wanted > 3) THEN Ok_answer := false;

      IF NOT Ok_answer THEN Invalid_answer;

  UNTIL Ok_answer = true;

  NRQuarters :=  Qtrs_wanted + (Years_wanted * 4);

    IF (Qtrs_wanted + (Years_wanted * 4) = 0) THEN
    BEGIN
    Color ( white, red);
    Blankline (24, 1);
    WRITE ('You have chosen 0 yrs & 0 qtrs.  Returning to selection menu.');
    DELAY (2000);
    Color ( white, blue);
    Blankline (24, 1);
    EXIT;  { Exit from ask for years procedure }
    END;  { if NRquarters = 0 }

  Calculations;

END; {Ask_for_years}


{.pa}
{*****************************************************************************
*****************************************************************************}


PROCEDURE Calculations;

VAR

  Temp,
  Temporary: REAL;
  TT,
  LL,
  L: INTEGER;
  AA: CHAR;
  Data_file: TEXT;
  Transfers: Two_Dim_Real;

BEGIN

  Toomany:=false;
  Blankline (24, 1);
  WRITE ('                    Beginning calculations.                  ');
  QtrCount:= 0; { count of years that model is running for }

    FOR TT := 12 DOWNTO 1 DO
    BEGIN

      FOR AA := 'A' TO 'H' DO
      BEGIN

        FOR LL := 1 TO 16 DO
        BEGIN
        Temp_billet_node [TT, AA, LL] := Billet_node [TT, AA, LL];
        END; { LL loop }

      END; { AA loop }

    END; { TT loop }

    REPEAT
    QtrCount := QtrCount + 1;

      FOR T:= 12 DOWNTO 2 DO
```

```
  BEGIN  (T Loop)

    FOR A:= 'A' TO 'H' DO
    BEGIN (A loop - mathematical calculations)

    L:= 1;  (Length of billet nodes in quarters)

      REPEAT
      Temp_Billet_node [ T, A, L ] := Temp_Billet_node [ T, A, L+1];
      L := L + 1;
      UNTIL L = 16 ;

    Transfers [ T-1, A ] :=
        (Temp_Billet_Node [ T-1, 'A' , 1 ] * Transfer_Path [T-1,'A',A]) +
        (Temp_Billet_Node [ T-1, 'B' , 1 ] * Transfer_Path [T-1,'B',A]) +
        (Temp_Billet_Node [ T-1, 'C' , 1 ] * Transfer_Path [T-1,'C',A]) +
        (Temp_Billet_Node [ T-1, 'D' , 1 ] * Transfer_Path [T-1,'D',A]) +
        (Temp_Billet_Node [ T-1, 'E' , 1 ] * Transfer_Path [T-1,'E',A]) +
        (Temp_Billet_Node [ T-1, 'F' , 1 ] * Transfer_Path [T-1,'F',A]) +
        (Temp_Billet_Node [ T-1, 'G' , 1 ] * Transfer_Path [T-1,'G',A]);

    Temporary :=   Temp_Billet_Node [ T, A, Billet_Length [ T, A ]];
    Temp_Billet_Node [ T, A, Billet_Length [ T, A ]]:=
        Transfers [T-1, A] + Temporary;
    Final_totals := false;
    Billet_totals (T,A);

      IF Temp_billet_total [ T, A ] > Hilimit [ T, A ] THEN
      BEGIN
      Violation := true;
      High_warning;

        IF Stop_calc_T THEN EXIT;

      END; (if high warning)

      IF Temp_billet_total [ T, A ] < Lolimit [ T, A ] THEN
      BEGIN
      Violation := true;
      Low_warning;

        IF Stop_calc_T THEN EXIT;

      END; ( if low warning )

    END;   ( A Loop )

  END; (T Loop and mathematical calculations)

UNTIL (QtrCount = NRQuarters) OR Toomany;

  IF NOT Violation THEN
  BEGIN
  TempCount := TempCount + QtrCount;
  YrCount := TRUNC (TempCount/4);

    IF ((YrCount * 4) > Tempcount) THEN YrCount := YrCount - 1;

  Quarter := TempCount - (YrCount * 4);


    FOR TT := 12 DOWNTO 1 DO
    BEGIN

      FOR AA := 'A' TO 'H' DO
      BEGIN

        FOR LL := 1 TO 16 DO
```

```
            BEGIN
            Billet_node [TT, AA, LL] := Temp_Billet_node [TT, AA, LL];
            END; { LL loop }

          END; { AA loop }

        END; { TT loop }

      Final_totals := true;
      All_billet_totals;

      END; { violations if }

END;  {calculations}
```

APPENDIX H

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "TOTALS.PAS"
{##########################################################################
##########################################################################}

{ This file contains the following procedures

    All_billet_totals
    BilTet_totals
    High_warning
    Low_warning

}


PROCEDURE All_billet_totals;

{ This procedure provides the mechanics for totalling
  all of the real billet nodes }

VAR

  K: CHAR;

BEGIN

  IF Final_totals THEN
  BEGIN

    FOR X:= 12 DOWNTO 1 DO
    BEGIN  { Activity Loop}

      FOR K:='A' TO 'H' DO
      BEGIN
      Billet_Total [ X, K ] := Billet_node [ X, K, 1 ] +
            Billet_node [ X, K, 2 ] + Billet_node [ X, K, 3 ] +
            Billet_node [ X, K, 4 ] + Billet_node [ X, K, 5 ] +
            Billet_node [ X, K, 6 ] + Billet_node [ X, K, 7 ] +
            Billet_node [ X, K, 8 ] + Billet_node [ X, K, 9 ] +
            Billet_node [ X, K, 10 ] + Billet_node [ X, K, 11 ] +
            Billet_node [ X, K, 12 ] + Billet_node [ X, K, 13 ] +
            Billet_node [ X, K, 14 ] + Billet_node [ X, K, 15 ] +
            Billet_node [ X, K, 16 ];
      END;  { K loop billet total }

    END; { X loop billet total }

  END; { if all_total is true }

END; {all_billet_totals }

{##########################################################################
##########################################################################}

{.pa}

PROCEDURE Billet_totals;

{ This procedure provides the mechanics for both all
  temporary billet_totals and individual totals of the billet nodes }

VAR
```

```
  K: CHAR;

BEGIN

  IF Final_totals THEN
  BEGIN

    FOR X:= 12 DOWNTO 1 DO
    BEGIN  ( Activity Loop)

       FOR K:='A' TO 'H' DO
       BEGIN
       Billet_Total [ X, K ] := Billet_node [ X, K, 1 ] +
             Billet_node [ X, K, 2 ] + Billet_node [ X, K, 3 ] +
             Billet_node [ X, K, 4 ] + Billet_node [ X, K, 5 ] +
             Billet_node [ X, K, 6 ] + Billet_node [ X, K, 7 ] +
             Billet_node [ X, K, 8 ] + Billet_node [ X, K, 9 ] +
             Billet_node [ X, K, 10 ] + Billet_node [ X, K, 11 ] +
             Billet_node [ X, K, 12 ] + Billet_node [ X, K, 13 ] +
             Billet_node [ X, K, 14 ] + Billet_node [ X, K, 15 ] +
             Billet_node [ X, K, 16 ];
       END;  ( K loop billet total )

     END; ( X lool billet total )

   END ( if all_total is true )

   ELSE
   BEGIN
   Temp_Billet_Total [ T, A ] := Temp_billet_node [ T, A, 1 ] +
          Temp_billet_node [ T, A, 2 ] + Temp_billet_node [ T, A, 3 ] +
          Temp_billet_node [ T, A, 4 ] + Temp_billet_node [ T, A, 5 ] +
          Temp_billet_node [ T, A, 6 ] + Temp_billet_node [ T, A, 7 ] +
          Temp_billet_node [ T, A, 8 ] + Temp_billet_node [ T, A, 9 ] +
          Temp_billet_node [ T, A, 10 ] + Temp_billet_node [ T, A, 11 ] +
          Temp_billet_node [ T, A, 12 ] + Temp_billet_node [ T, A, 13 ] +
          Temp_billet_node [ T, A, 14 ] + Temp_billet_node [ T, A, 15 ] +
          Temp_billet_node [ T, A, 16 ];
   END;   (else )

END; (totals )

(*****************************************************************************
*****************************************************************************)

(.pa)

PROCEDURE High_warning;

{ This procedure provides the displays and mechanics for
  displaying the high warning message and post warning
  choice display }

VAR

  Finish_high,
  Ok_activity: BOOLEAN;
  Tottempcount,
  Totyrcount,
  Totqtrcount,
  Totquarter,
  Qcount,
  TT,
  LL,
  L: INTEGER;
  Temp : REAL;
  AA,
```

```
  K: CHAR;
  Data_file: TEXT;

BEGIN

  Tottempcount := Tempcount + QtrCount;
  TotYrCount := TRUNC (TotTempcount/4);

      IF ((TotYrCount * 4) > Tottempcount ) THEN TotYrCount := TotYrCount-1;

  TotQuarter := TotTempcount - (TotYrCount * 4);

    REPEAT
    Clearscreen;
    Color ( white, red);
    GotoRC (2, 10);
    WRITELN ('After  ',TotYrcount,'  year(s) and ',TotQuarter,'  quarter(s), ');
    GotoRC (4, 10);
    WRITELN ('The constraint for ',Tour [ T ],' ',Activity [ A ],
          ' has been exceeded.');
    GotoRC (6, 10);
    WRITELN ('There are ',ROUND (Temp_billet_total [ T, A ]),' officers there now.');
    GotoRC (8, 10);
    WRITELN ('The high limit is ', ROUND (Hilimit [ T, A ]), ' ');
    Color ( white, blue);
    GotoRC (10, 1);
    WRITELN ('   What do you desire to do now?');
    WRITELN ('');
    WRITELN ('    0.  Abort simulation. Return to selection menu.');
    WRITELN ('    1.  Ignore limits and continue simulation.');
    WRITELN ('    2.  Back up one quarter in the simulation.');

      REPEAT  (until answer correct)
      GotoRC (10, 34);
      READ (kbd,  Choice ); (read ASSIGNMENT letter)
      First := '0'; Last := '2'; Ok_choice := true;
      Correct_choice ( Choice, First, Last, Ok_choice);
      Ok_activity := Ok_choice;
      UNTIL  Ok_activity  = true;

      IF Choice = '0' THEN
      BEGIN

        FOR TT := 12 DOWNTO 1 DO
        BEGIN

          FOR AA := 'A' TO 'H' DO
          BEGIN

            FOR LL := 1 TO 16 DO
            BEGIN
            Temp_billet_node [TT, AA, LL] := Billet_node [TT, AA, LL];
            Temp_billet_total [ TT, AA ] := Billet_total [ TT, AA ];
            END; { LL loop )

          END; ( AA loop )

        END; ( TT loop )

      Stop_calc_T := true;
      Toomany := true;
      Violation := false;
      END;  { if choice is zero / abort )

      IF Choice = '1' THEN
      BEGIN
      GotoRC (16, 1);
      WRITE (' Setting ', Tour [ T ], Activity [ A ],' high limit to 9999,');
```

```
      GotoRC (18, 1);
      WRITE ('         and continuing calculations!');
      Hillmit [ T, A ] := 9999;
      Violation := false;
      Stop_calc_T := false;
      END; { if choice was 1)

      IF Choice = '2' THEN
      BEGIN
      NRQuarters := QtrCount - 1;

        FOR TT := 12 DOWNTO 1 DO
        BEGIN

          FOR AA := 'A' TO 'H' DO
          BEGIN

            FOR LL := 1 TO 16 DO
            BEGIN
            Temp_billet_node [ TT, AA, LL ] := Billet_node [ TT, AA, LL ];
            Temp_billet_total [ TT, AA ] := Billet_total [ TT, AA ]
            END; { LL loop )

          END; { AA loop )

        END; { TT loop )

        IF NRQuarters = 0 THEN
        BEGIN
        Color ( white, red);
        Blankline (24, 1);
        WRITE ('One less quarter is 0 years and 0 quarters new calculations.');
        DELAY (2000);
        Blankline (24, 1);
        WRITE ('       Returning to selection menu.');
        DELAY (2000);
        Color ( white, blue);
        Blankline (24, 1);
        Stop_calc_T := true;
        END

        ELSE
        BEGIN
        GotoRC (16, 1);
        WRITELN ('  After completing calculations for one less quarter');
        WRITE ('   the model will return you to the selection menu.');
        DELAY (2000);
        Violation := false;
        Stop_calc_T := true;
        Calculations;

        END; { If NRQuarters = 0)

      END; { If choice was 2 )

    Finish_high := true;

    UNTIL Finish_high = true;

END;  { High warning )

(*****************************************************************************
*****************************************************************************)

(.pa)

PROCEDURE Low_warning;
```

```
{ This procedure provides the displays and mechanics for
  displaying the low warning message and post warning
  choice display }

VAR

  Finish_low,
  Ok_activity: BOOLEAN;
  Tottempcount,
  Totyrcount,
  Totqtrcount,
  Totquarter,
  Qcount,
  TT,
  LL,
  L: INTEGER;
  Temp : REAL;
  AA,
  K: CHAR;
  Data_file: TEXT;

BEGIN

  Tottempcount := Tempcount + QtrCount;
  TotYrCount := TRUNC (TotTempcount/4);

      IF ((TotYrCount * 4) > Tottempcount ) THEN TotYrCount := TotYrCount-1;

  TotQuarter := TotTempcount - (TotYrCount * 4);

    REPEAT
    Clearscreen;
    Color ( white, red);
    GotoRC (2, 10);
    WRITELN ('After  ',TotYrcount,'  year(s) and ',TotQuarter,'  quarter(s), ');
    GotoRC (4, 10);
    WRITELN ('The constraint for ',Tour [ T ],' ',Activity [ A ],
          ' has been violated.');
    GotoRC (6, 10);
    WRITELN ('There are ',ROUND (Temp_billet_total [ T, A ]),' officers there now.');
    GotoRC (8, 10);
    WRITELN ('The low limit is ', ROUND (Lolimit [ T, A ]), ' ');
    Color ( white, blue);
    GotoRC (10, 1);
    WRITELN ('  What do you desire to do now?');
    WRITELN ('');
    WRITELN ('   0.  Abort simulation. Return to selection menu.');
    WRITELN ('   1.  Ignore limits and continue simulation.');
    WRITELN ('   2.  Back up one quarter in the simulation.');

      REPEAT  (until answer correct)
      GotoRC (10, 34);
      READ (kbd,  Choice ); (read ASSIGNMENT letter)
      First := '0'; Last := '2'; Ok_choice := true;
      Correct_choice ( Choice, First, Last, Ok_choice);
      Ok_activity := Ok_choice;
      UNTIL  Ok_activity  = true;

      IF Choice = '0' THEN
      BEGIN

        FOR TT := 12 DOWNTO 1 DO
        BEGIN

          FOR AA := 'A' TO 'H' DO
          BEGIN

            FOR LL := 1 TO 16 DO
```

```
      BEGIN
      Temp_billet_node [TT, AA, LL] := Billet_node [TT, AA, LL];
      Temp_billet_total [ TT, AA ] := Billet_total [ TT, AA ];
      END; { LL loop }

    END; { AA loop }

  END; { TT loop }

Stop_calc_T := true;
Toofew := true;
Violation := false;
END;  { if choice is zero / abort }

IF Choice = '1' THEN
BEGIN
GotoRC (16, 1);
WRITE (' Setting ', Tour [ T ], Activity [ A ],' low limit to  0,');
GotoRC (18, 1);
WRITE ('         and continuing calculations!');
Lollmit [ T, A ] := 0;
Violation := false;
Stop_calc_T := false;
END; { if choice was 1}

IF Choice = '2' THEN
BEGIN
NRQuarters := QtrCount - 1;

  FOR TT := 12 DOWNTO 1 DO
  BEGIN

    FOR AA := 'A' TO 'H' DO
    BEGIN

      FOR LL := 1 TO 16 DO
      BEGIN
      Temp_billet_node [ TT, AA, LL ] := Billet_node [ TT, AA, LL ];
      Temp_billet_total [ TT, AA ] := Billet_total [ TT, AA ]
      END; { LL loop }

    END; { AA loop }

  END; { TT loop }

  IF NRQuarters = 0 THEN
  BEGIN
  Color ( white, red);
  Blankline (24, 1);
  WRITE ('One less quarter is 0 years and 0 quarters new calculations.');
  DELAY (2000);
  Blankline (24, 1);
  WRITE ('       Returning to selection menu.');
  DELAY (2000);
  Color ( white, blue);
  Blankline (24, 1);
  Stop_calc_T := true;
  END

  ELSE
  BEGIN
  GotoRC (16, 1);
  WRITELN ('  After completing calculations for one less quarter');
  WRITE ('   the model will return you to the selection menu.');
  DELAY (2000);
  Violation := false;
  Stop_calc_T := true;
  Calculations;
```

```
        END; { If NRQuarters = 0}
      END; { if choice was 2 }
    Finish_low := true;
   UNTIL Finish_low = true;
END;  { Low warning }
```

APPENDIX I

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "CH6DATA.PAS"
(##############################################################################
##############################################################################)

{ This file contains the following procedures

      Changes
      Change_accessions
      Change_arcs
      Change_length
      Change_hilimits
      Change_lolimits
 }

PROCEDURE Changes;

{ This procedure provides  the display and response evaluation
  for changing data }

BEGIN

  Finished := false;

    IF NOT (POS ('.',Nodedata) = 0) THEN Strip (Nodedata);
    IF NOT (POS ('.',Arcdata) = 0) THEN Strip (Arcdata);
    IF NOT (POS ('.',Lengdata) = 0) THEN Strip (Lengdata);
    IF NOT (POS ('.',Hilidata) = 0) THEN Strip (Hilidata);
    IF NOT (POS ('.',Lolidata) = 0) THEN Strip (Lolidata);

  REPEAT  (until finished true )
  ClearScreen;
  WRITELN;
  WRITELN ('     ');
  WRITELN ('       Which data do you desire to change?');
  WRITELN ('    ');
  WRITELN ('     Input data                                  Data file ');
  GotoRC (8, 1);
  WRITELN ('     0. No changes/finished changes. ');
  WRITELN ('     1. Number of officer accessions.');
  WRITELN ('     2. Transfer path percentages:              ', Arcdata  );
  WRITELN ('     3. Assignment tour lengths:                ', Lengdata);
  WRITELN ('     4. High limits:                            ', Hilidata );
  WRITELN ('     5. Low limits:                             ', Lolldata );
  WRITELN;WRITELN;

    REPEAT ( until choice correct )
    GotoRC (16,1);
    WRITE ('   Type your selection [0,1,2,3,4,5]: ');
    READ (Kbd, Cholce); (read input into var selection)
    First := '0';  Last := '5'; Ok_choice := true;  (inputs for correct_choice  )
    Correct_Choice (Cholce, Flrst, Last, Ok_choice);      (        procedure.         )
    UNTIL OK_choice = true;

    IF NOT Ok_choice THEN Changes;
    IF Choice = '0' THEN  Finished := true;
    IF Choice = '1' THEN  Change_accessions;
    IF Choice = '2' THEN  Change_arcs;
    IF Choice = '3' THEN  Change_length;
    IF Choice = '4' THEN  Change_hilimits;
    IF Choice = '5' THEN  Change_lolimits;
```

```
  UNTIL Finished = true;

END;  ( Changes )


{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}

{.pa}


PROCEDURE Change_arcs;

{ This procedure provides the display and mechanics
  to change transfer path percentages }

CONST

  Maxpercent = 100;

VAR

  Ok_selection,
  Ok_pct,
  Quit,
  Ok_total,
  Ok_tour,
  Ok_activity,
  Finished_chg,
  Finished_path: BOOLEAN;
  ZZ : CHAR;
  Path_total,
  P : REAL;
  R,
  RR,
  Total : INTEGER;

BEGIN

  Ok_pct := false;
  Ok_tour := false;
  Ok_activity := false;
  Ok_total := true;
  Quit := false;
  Finished_path := false;
  Finished_chg := false;

  REPEAT (until finished changes)

    REPEAT { until selection ok }
    Ok_selection := true;
    Chg_var := 'Which assignment do you wish to change the path % from?';
    Clearscreen;
    Dchoices;
    Clearscreen;

      IF (T = 12) THEN
      BEGIN
      Ok_selection := false;
      Color ( white, red);
      Blankline (24,1);
      WRITE (' There are no transfer paths from Twelfth Tour assignments.');
      DELAY (2000);
      Color ( white, blue);
      Blankline (24,1);
      END; { if tour 12 is chosen)
```

```
    IF (A in ['H','h']) THEN
    BEGIN
    Ok_selection := false;
    Color ( white, red);
    Blankline (24,1);
    WRITE (' There are no transfer paths from Separation assignments.');
    DELAY (2000);
    Color ( white, blue);
    Blankline (24,1);
    END; { if tour 12 is chosen)

  UNTIL Ok_selection = true;

Path_total := 0.0;

  FOR K:='A' TO 'H' DO
  BEGIN
  Path_total := Path_total + Transfer_path [ T, A, K ];
  END; ( path total)

From_pathscrn;
GotoRC(10,5);
WRITELN('   FROM');
GotoRC(11,2);
WRITELN(Tour [ T ],Activity [ A ]);
R := 1;
RR := 2;

  FOR ZZ := 'A' TO 'H' DO
  BEGIN
  GotoRC ( R, 40);
  WRITELN ('TO ',ZZ,': ', Tour [ T+1 ], Activity [ ZZ ]);
  GotoRC (RR, 42);
  WRITELN('',(Transfer_path[T,A,ZZ]*100):4:0,'%');
  R := R + 3;
  RR := RR + 3;
  END;

GotoRC (16,2);
WRITE ('Percent Total');
GotoRC (17,1);
WRITE   ('   ',(Path_total*100):4:0,' %');
Path_total := 0.0;

  REPEAT ( until Ok_total true )

    REPEAT  ( until quit with current path breakout)

      REPEAT (until ok_choice )
      GotoRC (24, 1);
      WRITE ('Which path # do you wish to change?(A-H)or(Q to quit)        ');
      GotoRC (24, 55);
      READ (Kbd, K);
      K := UPCASE (K);

        IF K in ['A','B','C','D','E','F','G','H','Q'] THEN Ok_choice := true
        ELSE
        BEGIN
        Ok_choice := false;
        Invalid_answer;
        END; { If K correct choice )

      UNTIL Ok_choice = true;

    Ok_choice := false;

      IF K = 'Q' THEN Quit:= true;
```

```
IF NOT (Quit) THEN
BEGIN

  REPEAT ( until Ok_pct true )

  IF K = 'A' THEN GotoRC (2, 42);
  IF K = 'B' THEN GotoRC (5, 42);
  IF K = 'C' THEN GotoRC (8, 42);
  IF K = 'D' THEN GotoRC (11, 42);
  IF K = 'E' THEN GotoRC (14, 42);
  IF K = 'F' THEN GotoRC (17, 42);
  IF K = 'G' THEN GotoRC (20, 42);
  IF K = 'H' THEN GotoRC (23, 42);

  {$I-} READ ( P ) {$I+} ; {read percent number but first cancel I/O }
                           { checking to prevent error if non real }
                      { key is struck accidentally. Then turn back on }
  Ok_pct := (IORESULT = 0);{TURBO PASCAL I/O error check.  If input }
                           { was in fact correct i.e. a real then }
                           { the TB function IORESULT will return a 0 }

  IF  (P > 100) OR (P < 0) THEN Ok_pct := false;

  IF Ok_pct AND (Billet_length [ T+1, K ] = 0) THEN
  BEGIN
  GotoRC (24, 1);
  Color ( white, red );
  Blankline (24, 1);
  WRITE ('  You are sending officers to an assignment with a length of 0 ');
  DELAY (2000);
  Blankline (24, 1);
  WRITE (' Do not forget to change the tour length of that assignment.');
  DELAY (2000);
  Color ( white, blue);
  Blankline (24, 1);
  END;

  IF NOT Ok_pct THEN Invalid_answer;

  UNTIL Ok_pct = true;

Ok_pct := false;
Transfer_path [ T, A, K] := (P/100);

  IF K = 'A' THEN GotoRC (2, 42);
  IF K = 'B' THEN GotoRC (5, 42);
  IF K = 'C' THEN GotoRC (8, 42);
  IF K = 'D' THEN GotoRC (11, 42);
  IF K = 'E' THEN GotoRC (14, 42);
  IF K = 'F' THEN GotoRC (17, 42);
  IF K = 'G' THEN GotoRC (20, 42);
  IF K = 'H' THEN GotoRC (23, 42);

Color ( blue, white);
WRITE ('',(Transfer_path [T,A,K]*100):4:0,' %');
Path_total := 0.0;

  FOR K:='A' TO 'H' DO
  BEGIN
  Path_total := Path_total + Transfer_path [ T, A, K ];
  END; ( path total)

GotoRC (17, 1);
WRITE   ('   ',(Path_total*100):4:0,' %');
Color ( white, blue );

END;  ( not quit if)
```

```
      UNTIL Quit = true;

      IF Quit THEN
      BEGIN
      Ok_total := true;
      Path_total := 0.0;

        FOR K:='A' TO 'H' DO
        BEGIN
        Path_total := Path_total + Transfer_path [ T, A, K ];
        END; { path total)

      Total := ROUND (Path_total * 100);

        IF (Total = 0) OR (Total = 100)
            THEN Ok_total := true
        ELSE
        BEGIN
        Ok_total := false;
        Ok_pct := false;
        Ok_choice := false;
        Quit := false;
        Color ( white, red);
        GotoRC (24,1);
        Blankline (24, 1);
        WRITE ('The total of transfer percentages must equal 100 or 0');
        DELAY (2000);
        Blankline (24, 1);
        WRITE ('      They now sum to ',Total,'                          ');
        DELAY (2000);
        Blankline (24, 1);
        WRITE ('    You need to change another path percentage.      ');
        DELAY (2000);
        Color ( white, blue);
        Blankline (24, 1);
        GotoRC (23, 30);
        END; { if percent not equal to 100 )

      END; {if quit true)

    Quit := false;

    UNTIL Ok_total = true;

  Ok_total := false;

  Another_change;

    IF Answer in ['N','n'] THEN Finished_chg := true;

  UNTIL Finished_chg = true;

END; {Change_arcs)

(*****************************************************************************
*****************************************************************************)


(.pa)

PROCEDURE Change_length;

{ This procedure provides the displays and mechanics for
  changing assignment tour length }

VAR
  Finished_leng,
  Ok_tour,
```

```
  Ok_activity: Boolean;

BEGIN

  Ok_tour := false;
  Ok_activity := false;
  Finished_leng := false;

    REPEAT (until finished_leng true )
    Chg_var := 'Which assignment tour length do you wish to change?';
    Clearscreen;
    Dchoices;

      REPEAT ( until number entry is correct )
      Clearscreen;
      GotoRC (3, 1);
      WRITELN ('    The old ', Tour [ T ], Activity [ A ],' length is ',
           Billet_length [ T, A ]:0,' quarters');
      WRITELN;WRITELN;WRITELN;
      WRITE ('    How  many quarters long do you want this assignment now? ');
      {$I-}  READ ( New_Length [ T, A ])  {$I+}; (read new length number)
      Ok_answer := (IORESULT = 0);  { TURBO PASCAL I/O error check.  If input }
                                    { was in fact correct i.e. an integer then }
                                    { the TB function IORESULT will return a 0 }

        IF NOT Ok_answer THEN Wrong_answer;

      UNTIL Ok_answer = true;

    GotoRC (10, 5);
    WRITELN ('The new ', Tour [ T ],Activity [A],
                ' length is    ', New_Length [ T,A],'    quarters.');
    Billet_Length [ T, A ]:= New_Length [ T, A ];
    Another_change;

      IF Answer in ['Y','y'] THEN Finished_leng := false;
      IF Answer in ['N','n'] THEN Finished_leng := true;

    UNTIL Finished_leng  = true;

  Finished := false;

END; { Change_length }


{*****************************************************************************
*****************************************************************************}

{.pa}

PROCEDURE Change_hilimits;

{ This procedure provides the display and mechanics for
  making changes to the high limit data }

VAR

  Finished_high,
  Ok_tour,
  Ok_activity: Boolean;
  New_limit: Two_Dim_Real;

BEGIN

  Finished_high := false;
  Ok_tour := false;
  Ok_activity := false;
```

```
    REPEAT ( until finished_high is true )
    Chg_var := 'Which high limit do you wish to change?';
    Clearscreen;
    Dchoices;

      REPEAT ( until number entry is correct )
   Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
           Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
           Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
           Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
           Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
           Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
           Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
           Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
           Billet_node [ T, A, 16 ];
      Clearscreen;
      GotoRC (3, 1);
      WRITELN (' The old high limit for ', Tour [ T ], Activity [ A ],
               ' is ', Hilimit [ T, A ]:8:0,' officers.');
      WRITELN;
      WRITELN ('          There currently are ', Billet_total [ T, A ]:8:0,
                '   assigned.');
      New_limit [ T, A ] := Lolimit [ T, A];
      WRITELN;
      WRITE (' How  many officers do you want as the high limit now?  ');
      {$I-}  READLN ( New_Limit [ T, A ])  {$I+}; (read new length number)
      Ok_answer := (IORESULT = 0);  ( TURBO PASCAL I/O error check.  If input )
                                    ( was in fact correct i.e. an integer then )
                                    ( the TB function IORESULT will return a 0 )

        IF NOT Ok_answer THEN Wrong_answer;

      UNTIL Ok_answer = true;

    GotoRC (10,1);
    WRITELN (  '       The new  high limit for ', Tour [T],
             Activity [A],' is ', New_limit [ T, A ]:8:0,'   officers.');
    Hilimit [ T, A ]:=  New_limit [ T, A ];
    Another_change;

      IF Answer in ['N','n'] THEN Finished_high := true;

    UNTIL Finished_high = true;

END;    { Change_hilimits }


{*****************************************************************************
*****************************************************************************}


(.pa)

PROCEDURE Change_lolimits;

{ This procedure provides the display and mechanics for
  making changes to the low limit data }

VAR

  Finished_low,
  Ok_tour,
  Ok_activity: Boolean;
  New_limit: Two_Dim_Real;

BEGIN

  Finished_low := false;
```

```
  Ok_tour := false;
  Ok_activity := false;

    REPEAT (until finished_low is true)
    Chg_var := 'Which low limit do you wish to change?';
    Clearscreen;
    Bchoices;

       REPEAT ( until number entry is correct )
   Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
            Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
            Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
            Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
            Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
            Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
            Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
            Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
            Billet_node [ T, A, 16 ];

       Clearscreen;
       GotoRC (3, 1);
       WRITELN ('The old low limit for ', Tour [ T ], Activity [ A ],
                ' is ', Lolimit [ T, A ]:8:0,' officers.');
       WRITELN;
       WRITELN ('        There are currently ', Billet_total [ T, A ]:8:0,
                                '    assigned.');
       New_limit [ T , A ] := Lolimit [ T, A ];
       WRITELN;
       WRITE ('   How  many officers do you want as the low limit now?  ');
       {$I-}  READLN ( New_Limit [ T, A ])  {$I+}; (read new length number)
       Ok_answer := (IORESULT = 0);  { TURBO PASCAL I/O error check.  If input }
                                     { was in fact correct i.e. an integer then }
                                     { the TB function IORESULT will return a 0 }

         IF NOT Ok_answer THEN Wrong_answer;

      UNTIL Ok_answer = true;

    GotoRC (10,1);
    WRITELN ('    The new ',Tour [T], Activity [A],
             ' low limit is ', New_limit [ T, A ]:8:0,'   officers.');
    Lolimit [ T, A ]:=  New_limit [ T, A ];
    Another_change;

      IF Answer in ['N','n'] THEN Finished_low := true;

    UNTIL Finished_low = true;

END;  {Change_lolimit}

{*****************************************************************************
*****************************************************************************}

(.pa)

PROCEDURE Change_accessions;

{ This procedure provides the display and mechanics for
  making changes to the accessions }

VAR

  Finished_acc: BOOLEAN;

BEGIN

  REPEAT
  Clearscreen;
```

```
  GotoRC (12, 1);
  WRITELN ('     Currently the number of accessions each quarter is');
  WRITELN;
  WRITELN ('                    ',ROUND(Accessions),
                                  ' officers.');

    REPEAT
    GotoRC (16, 1);
    WRITE ('      The new number of accessions each quarter is:     ');
    GotoRC (17, 1);
    WRITE ('                  Press <CR> for no change.');
    GotoRC (16, 55);
    {$I-}  READ ( Accessions)  {$I+}; (read new length number)
    Ok_answer := (IORESULT = 0);  { TURBO PASCAL I/O error check.  If input }
                                  { was in fact correct i.e. an integer then }
                                  { the TB function IORESULT will return a 0 }

      IF NOT Ok_answer THEN Wrong_answer;

    UNTIL Ok_answer = true;

    IF Ok_answer THEN Finished_acc:= true;

  GotoRC (16, 53);
  WRITE (ROUND(Accessions),'   officers.');
  Delay (1500);

  UNTIL Finished_acc = true;

END;  { change accessions}
```

APPENDIX J

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "SCREENS.PAS"
(*****************************************************************************
*****************************************************************************)

{ This file contains the following procedures

    Dchoices
    Dchanges
    From_pathscrn
    To_pathscrn
}


PROCEDURE Dchoices;

{  This procedure provides the display screen and answer evaluation
   for the choices indicated }

VAR

  Ok_range,
  Ok_tour,
  Ok_activity: Boolean;

BEGIN

  Ok_tour := false;
  Ok_activity := false;
  GotoRC (1, 1);
  WRITELN ('   ',Chg_var);
  WRITELN;WRITELN;
  WRITELN ('                TOUR                ACTIVITY');
  WRITELN;
  WRITELN ('                1.  FIRST           A.  PROFESSIONAL TRNG ');
  WRITELN ('                2.  SECOND          B.  PROFESSIONAL EDUC ');
  WRITELN ('                3.  THIRD           C.  WASHINGTON DC     ');
  WRITELN ('                4.  FOURTH          D.  SHORE CONUS       ');
  WRITELN ('                5.  FIFTH           E.  FLEET UNIT        ');
  WRITELN ('                6.  SIXTH           F.  AFLOAT STAFF      ');
  WRITELN ('                7.  SEVENTH         G.  SHORE OUTUS       ');
  WRITELN ('                8.  EIGHTH          H.  SEPARATION        ');
  WRITELN ('                9.  NINTH');
  WRITELN ('               10.  TENTH');
  WRITELN ('               11.  ELEVENTH');
  WRITELN ('               12.  TWELFTH');
  GotoRC (22,5);
  WRITELN ('   TOUR:    (type number<CR>)  ACTIVITY:    (type letter)');

    REPEAT { UntiI Ok_tour true }

      REPEAT { UntiI Ok_activity true }

        IF NOT Ok_tour OR NOT Ok_range THEN
        BEGIN
        GotoRC (22, 14);
        {$I-} READ ( T ) {$I+} ; {read TOUR number but first cancel I/O }
                               { checking to prevent error if non integer }
                               { key is struck accidentally. Then turn back on }
        Ok_tour := (IORESULT = 0);  { TURBO PASCAL I/O error check.  If input }
                                    { was in fact correct i.e. an integer then }
                                    { the TB function IORESULT will return a 0 }
```

```
        IF (T >= 1) AND (T <= 12) THEN Ok_range := true;
        IF (T < 1) OR (T > 12) THEN Ok_range := false;
        IF NOT Ok_tour OR NOT Ok_range THEN Wrong_answer;

        END; { Ok_tour correct if }

      UNTIL (Ok_tour AND Ok_range) = true;

      IF NOT Ok_activity THEN
      BEGIN
      GotoRC (22, 46);
      READ (kbd,  A ); {read ASSIGNMENT letter}
      A := UPCASE ( A );
      First := 'A'; Last := 'H'; Ok_choice := true;
      Correct_choice ( A, First, Last, Ok_choice);
      Ok_activity := Ok_choice;
      END;   { if not ok activity }

    UNTIL (Ok_tour and Ok_activity ) = true;

  Clearscreen;

END; {choices}

{**************************************************************************
**************************************************************************}

{.pa}

PROCEDURE Dchanges;

{ This procedure provides the  choice menu display and
  also evaluates the responses }

BEGIN

  IF NOT (POS ('.',Nodedata) = 0) THEN Strip (Nodedata);
  IF NOT (POS ('.',Arcdata) = 0) THEN Strip (Arcdata);
  IF NOT (POS ('.',Lengdata) = 0) THEN Strip (Lengdata);
  IF NOT (POS ('.',Hilidata) = 0) THEN Strip (Hilidata);
  IF NOT (POS ('.',Lolidata) = 0) THEN Strip (Lolidata);

  Clearscreen;
  GotoRC (3, 5);
  WRITELN (Chg_var);
  WRITELN;
  WRITELN ('     Data:                                    Data files:');
  WRITELN;
  GotoRC (8, 1);
  WRITELN ('     0. ',Choice_zero);
  WRITELN ('     1. Number of officers at each assignment: ', Nodedata );
  WRITELN ('     2. Transfer path percentages:             ', Arcdata  );
  WRITELN ('     3. Assignment tour lengths:               ', Lengdata);
  WRITELN ('     4. High limits:                           ', Hilidata );
  WRITELN ('     5. Low limits:                            ', Lolidata );
  WRITELN;WRITELN;

    REPEAT {until choice correct }
    GotoRC (16,1);
    WRITE ('   Type your selection [0,1,2,3,4,5]: ');
    READ (Kbd, Choice); {read input into var selection}
    First := '0';  Last := '5'; Ok_choice := true;  {inputs for correct_choice  }
    Correct_Choice (Choice, First, Last, Ok_choice);      {        procedure.        }
    UNTIL Ok_choice = true;

END; {Dchanges}
```

```
{##############################################################################
##############################################################################}

{.pa}

PROCEDURE From_pathscrn;

{ This procedure simply draws the illustration on the screen }

BEGIN

  Clearscreen;
  WRITELN('                                /---          ');
  WRITELN('                               /             ');
  WRITELN('                              /              ');
  WRITELN('                             /------         ');
  WRITELN('                            /                ');
  WRITELN('                           /                 ');
  WRITELN('                          /---------         ');
  WRITELN('                         /                  ');
  WRITELN('                        /                   ');
  WRITELN('                       /------------        ');
  WRITELN('                      <                     ');
  WRITELN('                       \                    ');
  WRITELN('                        \-----------        ');
  WRITELN('                         \                  ');
  WRITELN('                          \                 ');
  WRITELN('                           \--------        ');
  WRITELN('                            \               ');
  WRITELN('                             \              ');
  WRITELN('                              \-----        ');
  WRITELN('                               \             ');
  WRITELN('                                \            ');
  WRITELN('                                 \--          ');

END; { From_pathscrn }

{##############################################################################
##############################################################################}

{.pa}

PROCEDURE To_pathscrn;

{ This procedure simply draws the illustration on the screen }

BEGIN

  Clearscreen;
  WRITELN ('                         ---------\              ');
  WRITELN ('                                   \             ');
  WRITELN ('                                    \            ');
  WRITELN ('                         ------------\           ');
  WRITELN ('                                     \           ');
  WRITELN ('                                      \          ');
  WRITELN ('                         ---------------\        ');
  WRITELN ('                                        \        ');
  WRITELN ('                                         \       ');
  WRITELN ('                         -------------------\    ');
  WRITELN ('                                             >   ');
  WRITELN ('                                            /    ');
  WRITELN ('                         -----------------/      ');
  WRITELN ('                                         /       ');
  WRITELN ('                                        /        ');
  WRITELN ('                         --------------/         ');
  WRITELN ('                                      /          ');
  WRITELN ('                                     /           ');
  WRITELN ('                         -----------/            ');
```

APPENDIX K

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "DISPLAYS.PAS"
(##########################################################################
##########################################################################)

{  This file contains the following procedures

    Disp_assign
    Disp_tours
    Disp_paths_from
    Disp_paths_to
    Disp_billets


}


PROCEDURE Disp_Assign;

{ This procedure provides the activity v.s. all tours choices and displays }

VAR

  I: INTEGER;

BEGIN

    REPEAT   { until finished}
    Finished:= false;
    Clearscreen;
    Center (03, 'Which activity are you interested in?');
    WRITELN;WRITELN;WRITELN;
    WRITELN ('                    A.  Professional Training');
    WRITELN ('                    B.  Professional Education');
    WRITELN ('                    C.  Washington DC');
    WRITELN ('                    D.  Shore (CONUS)          ');
    WRITELN ('                    E.  Fleet Unit');
    WRITELN ('                    F.  Afloat Staff');
    WRITELN ('                    G.  Shore (OUTUS)');
    WRITELN ('                    H.  Separation');
    WRITELN;

      REPEAT {until choice correct }
      GotoRC (18,15);
      WRITE ('Type your choice: ');
      READ (Kbd, A); {read input into var selection}
      A := UPCASE (A);
      First := 'A'; Last := 'H'; Ok_choice := true;
      Correct_choice ( A, First, Last, Ok_choice);
      UNTIL OK_choice = true ;
      Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
            Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
            Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
            Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
            Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
            Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
            Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
            Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
            Billet_node [ T, A, 16 ];
    Clearscreen;
    WRITELN;
    WRITELN ('    For    ',Yrcount,'  year(s) and  ',
```

```
                    Quarter,' quarter(s) calculations.');
WRITELN;
WRITELN ('           For: ', Activity [ A ]);
GotoRC (6, 1);
WRITELN ('              TOURS.....');
WRITELN;
GotoRC (8, 10);
WRITELN ('FIRST    SECOND       THIRD    FOURTH     FIFTH     SIXTH');
GotoRC (14, 8);
WRITELN ('SEVENTH    EIGHTH       NINTH     TENTH  ELEVENTH   TWELFTH');
T:=1;
X:= 5;
GotoRC (10, X);
Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
       Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
       Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
       Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
       Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
       Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
       Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
       Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
       Billet_node [ T, A, 16 ];
WRITE (ROUND (Billet_total [ T, A ]):10);

  FOR T := 2 TO 6 DO
  BEGIN
  X := X + 10;
  GotoRC (10, X);
  Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
       Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
       Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
       Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
       Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
       Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
       Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
       Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
       Billet_node [ T, A, 16 ];
  WRITE (ROUND (Billet_total [ T, A ]):10);
  END;

T:=7;
X:= 5;
GotoRC (16, X);
Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
       Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
       Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
       Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
       Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
       Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
       Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
       Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
       Billet_node [ T, A, 16 ];
WRITE (ROUND (Billet_total [ T, A ]):10);

  FOR T := 8 TO 12  DO
  BEGIN
  X := X + 10;
  GotoRC (16, X);
  Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
       Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
       Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
       Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
       Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
       Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
       Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
       Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
       Billet_node [ T, A, 16 ];
  WRITE (ROUND (Billet_total [ T, A ]):10);
```

```
      END;

      REPEAT ( until answer correct )
      Center (19, 'Do you desire to see another activity breakout? (Y/N) ');
      READ (Kbd, Answer);
      Ok_answer := true;
      Correct_answer (Answer, Ok_answer);
      UNTIL Ok_answer = true;

      IF Answer in [ 'Y','y'] THEN
      BEGIN
      Clearscreen;
      Disp_Assign;
      END;

      IF Answer in [ 'N','n' ] THEN Finished := true;

    UNTIL Finished = true;

  Clearscreen;

END; (Disp_Assign)

{*************************************************************************
*************************************************************************)

{.pa}

PROCEDURE Disp_Tours;

{ This procedure provides the tour v.s. all activities choices and displays }

VAR

  Ok_range,
  Ok_tour,
  Ok_activity: Boolean;

BEGIN

  Ok_range := false;
  Ok_tour := false;
  Ok_activity := false;

    REPEAT ( until finished )
    Clearscreen;
    GotoRC (3, 5);
    CENTER (3,'Which Tour do you wish to see displayed?');
    WRITELN;WRITELN;
    WRITELN ('                  TOUR                ');
    WRITELN;
    WRITELN ('                 1.  FIRST            ');
    WRITELN ('                 2.  SECOND           ');
    WRITELN ('                 3.  THIRD            ');
    WRITELN ('                 4.  FOURTH           ');
    WRITELN ('                 5.  FIFTH            ');
    WRITELN ('                 6.  SIXTH            ');
    WRITELN ('                 7.  SEVENTH          ');
    WRITELN ('                 8.  EIGHTH           ');
    WRITELN ('                 9.  NINTH');
    WRITELN ('                10.  TENTH');
    WRITELN ('                11.  ELEVENTH');
    WRITELN ('                12.  TWELFTH');
    GotoRC (22, 5);
    WRITELN ('   TOUR:      (type number<CR>)');

      REPEAT  ( Until Ok_tour and Ok_range both true)
      GotoRC (22, 14);
```

```
   {$I-} READ ( T ) {$I+} ; {read TOUR number but first cancel I/O }
                            { checking to prevent error if non integer }
                            { key is struck accidentally. Then turn back on }
   Ok_tour := (IORESULT = 0);  { TURBO PASCAL I/O error check.  If input }
                               { was in fact correct i.e. an integer then }
                               { the TB function IORESULT will return a 0 }

     IF  (T >= 1) AND (T <= 12) THEN Ok_range := true;
     IF  (T < 1) OR (T > 12) THEN Ok_range := false;
     IF NOT Ok_tour OR NOT Ok_range THEN Wrong_answer;

   UNTIL (Ok_tour AND Ok_range) = true;
FOR A := 'A' TO 'H' DO
BEGIN
Billet_total [ T, A ] := Billet_node [ T, A, 1 ] +
        Billet_node [ T, A, 2 ] + Billet_node [ T, A, 3 ] +
        Billet_node [ T, A, 4 ] + Billet_node [ T, A, 5 ] +
        Billet_node [ T, A, 6 ] + Billet_node [ T, A, 7 ] +
        Billet_node [ T, A, 8 ] + Billet_node [ T, A, 9 ] +
        Billet_node [ T, A, 10 ] + Billet_node [ T, A, 11 ] +
        Billet_node [ T, A, 12 ] + Billet_node [ T, A, 13 ] +
        Billet_node [ T, A, 14 ] + Billet_node [ T, A, 15 ] +
        Billet_node [ T, A, 16 ];
END; {for loop}

Clearscreen;
WRITELN ('  For    ',Yrcount,'  year(s) and  ',Quarter,'  quarter(s) calculations. ');
WRITELN ('              For: ', Tour [ T ]);
WRITELN ('                   ');
WRITELN ('  Activity:                    Number of officers');
WRITELN ('                                                  ');
WRITELN ('  Professional Training  : ', ROUND (Billet_total [ T, 'A' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Professional Education : ', ROUND (Billet_total [ T, 'B' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Washington DC          : ', ROUND (Billet_total [ T, 'C' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Shore (CONUS)          : ', ROUND (Billet_total [ T, 'D' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Fleet Unit             : ', ROUND (Billet_total [ T, 'E' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Afloat Staff           : ', ROUND (Billet_total [ T, 'F' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Shore (OUTUS)          : ', ROUND (Billet_total [ T, 'G' ]):10);
WRITELN ('                                                  ');
WRITELN ('  Separation             : ', ROUND (Billet_total [ T, 'H' ]):10);
WRITELN ('                                                  ');
Tour_sum [ T ] := Billet_total [ T, 'A'] + Billet_total [ T, 'B'] +
                  Billet_total [ T, 'C'] + Billet_total [ T, 'D'] +
                  Billet_total [ T, 'E'] + Billet_total [ T, 'F'] +
                  Billet_total [ T, 'G'] + Billet_total [ T, 'H'];
WRITELN ('  Total officers         : ', ROUND (Tour_sum [ T ]):10);

   REPEAT { until answer correct }
   Blankline (24,1);
   WRITE ('Do you desire to see another tour breakout? (Y/N)  ');
   READ (Kbd, Answer);
   Ok_answer := true;
   Correct_answer (Answer, Ok_answer);
   UNTIL Ok_answer = true;

   IF Answer in [ 'Y','y'] THEN
   BEGIN
   Disp_tours;
   END;

   IF Answer in [ 'N','n' ] THEN Finished := true;
```

```
    UNTIL Finished = true;

  Clearscreen;

END; (Disp_Tours)

{##########################################################################
##########################################################################}

{.pa}

PROCEDURE Disp_paths_from;

{  This procedure displays the transfer path percentages FROM }

VAR

  Ok_selection,
  Ok_tour,
  Ok_activity: BOOLEAN;
  ZZ : CHAR;
  R,
  RR : INTEGER;

BEGIN

  Ok_tour := false;
  Ok_activity := false;
  Finished := false;

    REPEAT (until finished)

      REPEAT { until tour/activity selection ok }
      Ok_selection := true;
      Clearscreen;
      Chg_var := 'Which assignment do you wish to see the transfer paths from?';
      Dchoices;
      Clearscreen;

        IF (T = 12) THEN
        BEGIN
        Ok_selection := false;
        Color ( white, red);
        Blankline (24,1);
        WRITE (' There are no transfer paths from Twelfth Tour assignments.');
        DELAY (2000);
        Color ( white, blue);
        Blankline (24,1);
        END; { if tour 12 is chosen)

        IF A in ['H'] THEN
        BEGIN
        Ok_selection := false;
        Color ( white, red);
        Blankline (24,1);
        WRITE (' There are no transfer paths from Separation assignments.');
        DELAY (2000);
        Color ( white, blue);
        Blankline (24,1);
        END; { if tour 12 is chosen)

      UNTIL Ok_selection = true;

    From_pathscrn;
    GotoRC (6, 5);
    WRITELN ('  Display for ');
    GotoRC (7, 5);
    WRITELN (' ',YrCount,'  years and');
```

```
    WRITE ('     ',Quarter,' quarters');
    GotoRC (10, 5);
    WRITE ('   FROM');
    GotoRC(11, 2);
    WRITE (Tour  [ T ],Activity [ A ]);
    GotoRC (13, 6);
    WRITE (ROUND (Billet_node[T,A,1]),' officers');
    WRITE (' will be transferred.');
    R := 1;
    RR := 2;
    FOR ZZ := 'A' TO 'H' DO
    BEGIN
    GotoRC ( R, 40);
    WRITE ('TO ', Tour [ T+1 ], Activity [ ZZ ]);
    GotoRC (RR, 42);
    WRITE ('',(Transfer_path[T,A,ZZ]*100):4:0,'%   or ',
          ROUND ((Billet_node[T,A,1])*(Transfer_path[T,A,ZZ])),' officers');
    R := R + 3;
    RR := RR + 3;
    END;

    REPEAT (until answer correct )
    Blankline (24, 1);
    WRITE ('Do you desire to see another transfer path breakout? (Y/N) ');
    READ (Kbd, Answer);
    Correct_answer (Answer, Ok_answer);
    UNTIL Ok_answer = true;

  Ok_tour := false;    ( to prevent procedure from )
  Ok_activity := false;( repeating endlessly )

    IF Answer in ['N','n'] THEN Finished:= True;

  UNTIL Finished = true;

END; (Disp_paths_from)

{*************************************************************************
*************************************************************************}

(.pa)

PROCEDURE Disp_paths_to;

{ This procedure displays the transfer path percentages TO }

VAR

  Ok_selection,
  Ok_tour,
  Ok_activity: BOOLEAN;
  ZZ : CHAR;
  R,
  RR : INTEGER;
  Total: REAL;

BEGIN

  Ok_tour := false;
  Ok_activity := false;
  Finished := false;

    REPEAT (until finished)

      REPEAT ( until ok selection )
      Ok_selection := true;
      Clearscreen;
      Chg_var := 'Which assignment do you wish to see the transfer paths to?';
```

```
      Bchoices;
      Clearscreen;

        IF (T = 1) THEN
        BEGIN
        Ok_selection := false;
        Color ( white, red);
        Blankline (24,1);
        WRITE (' There are no transfer paths to First Tour assignments.');
        DELAY (2000);
        Color ( white, blue);
        Blankline (24, 1);
        END; ( if tour 12 is chosen)

      UNTIL Ok_selection = true;

    To_pathscrn;
    GotoRC (6, 55);
    WRITELN ('  Display for ');
    GotoRC (7, 55);
    WRITELN(YrCount,'  years and');
    GotoRC (8, 55);
    WRITE (Quarter,' quarters');
    GotoRC (10, 55);
    WRITELN ('  TO');
    GotoRC (11, 55);
    WRITELN (Tour  [ T ],Activity [ A ]);
    GotoRC (18, 45);
    WRITE ('NOTE: Percentages reflect the % of');
    GotoRC (19, 45);
    WRITE ('officers transferred OUT OF the');
    GotoRC (20, 45);
    WRITE ('indicated "FROM" assignment.');
    R := 1;
    RR := 2;
    Total := 0;

      FOR ZZ := 'A' TO 'G' DO
      BEGIN
      GotoRC ( R, 1);
      WRITELN ('FROM ', Tour [ T-1 ], Activity [ ZZ ]);
      GotoRC (RR, 1);
      WRITE ('',(Transfer_path[T-1,ZZ,A]*100):4:0,'%   or ',
            ROUND ((Billet_node[T-1,ZZ,1])*(Transfer_path[T-1,ZZ,A])),' officers');
      Total := Total + ((Billet_node[T-1,ZZ,1])*(Transfer_path[T-1,ZZ,A]));
      R := R + 3;
      RR := RR + 3;
      END;

    GotoRC (13, 53);
    WRITE (ROUND (Total), ' officers');

      REPEAT (until answer correct )
      Blankline (24,1);
      WRITE ('Do you desire to see another transfer path breakout? (Y/N) ');
      READ (Kbd, Answer);
      Correct_answer (Answer, Ok_answer);
      UNTIL OK_answer = true;

    Ok_tour := false;    ( to prevent procedure from )
    Ok_activity := false;( repeating endlessly )

      IF Answer in ['N','n'] THEN Finished:= True;

    UNTIL Finished = true;

END; ( Disp_paths_to )
```

```
{*************************************************************************
*************************************************************************}

{.pa}

PROCEDURE Disp_billets;

{ This procedure provides the billet by tour length choices and displays }

VAR

  Officer_totals: REAL;
  X : INTEGER;
  Finished,
  Ok_tour,
  Ok_activity: BOOLEAN;

BEGIN

  Ok_tour := false;
  Ok_activity := false;

    REPEAT { until finished }
    Clearscreen;
    Chg_var := 'Which assignment do you wish to see?';
    Dcholces;
    Clearscreen;
    WRITELN;WRITELN;
    WRITELN ('    After   ', Yrcount,' year(s)   ',
                          Quarter,' quarter(s)');
    WRITELN ('           ',Tour [T],' ',Activity [A]);
    WRITELN ('           Tour Length of ',
                            Billet_length [T,A],' quarters.');
    WRITELN ('    Officers assigned:');
    WRITELN;
    Officer_totals := 0;

      FOR X := 1 TO 16 DO
      BEGIN
      WRITELN ('     ',ROUND(Billet_node [ T, A, X ]),'      ', TLength [ X ]);
      Officer_totals := Officer_totals + Billet_node [ T, A, X ];
      END; { do loop}

    WRITELN;
    WRITELN (' The total number of officers assigned is ',
            ROUND (Officer_totals));

      REPEAT { until answer is correct }
      Blankline (24,1);
      WRITE ('Do you desire to see another assignment?  (Y/N)  ');
      READ (Kbd, Answer);
      Correct_answer (Answer, Ok_answer);
      UNTIL Ok_answer = true;

      IF Answer in ['N','n']  THEN Finished:=true;

    UNTIL Finished = true;

  Clearscreen;

END; { Disp_billet nodes }
```

APPENDIX L

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "DATADUMP.PAS"
(**************************************************************************
**************************************************************************)

{ This file contains the following procedures

   Node_dump
   Arc_dump
   Length_dump
   Hilim_dump
   Lolim_dump

}

PROCEDURE Node_dump;

{ This procedure provides the displays and mechanices for
  dumping the number of officers per assignment data
  from the program into a disk file }

VAR

   Ok_file: BOOLEAN;

BEGIN

   Clearscreen;
   Ok_file := false;

     IF NOT Replace THEN
     BEGIN

       REPEAT { until output filename is less than 8 letters}
       GotoRC (10,10);
       WRITELN ('Enter output filename (maximum of 8 letters)');
       Color ( red, white);
       GotoRC (13, 8);
       WRITELN (' UNLESS YOU DESIRE TO OVERWRITE PREVIOUS DATA, DO NOT USE ');
       Dirlist;
       Color ( white, blue);
       GotoRC (20, 13);
       READLN (Output_name);

         IF (Length (Output_name) > 8) THEN Wrong_answer

         ELSE
         BEGIN
         Ok_file := true;
         END; { if length of answer over 8 letters }

       UNTIL Ok_file = true;

     Ok_file := false;
     Color ( white, blue);
     END; { if replace false }

   Blankline (24, 1);
   WRITE (' Dumping billet node data to the file named : ', Output_name);
   T:= 1;
   Skip := ' ';
   Output_name := CONCAT (Output_name,'.nod');
```

```
   Nodedata := Output_name;
   Strip (Nodedata);
   ASSIGN (Data_file, Output_name);
   REWRITE (Data_file);

     FOR T := 1 TO 12 DO
     BEGIN

       FOR A := 'A' TO 'H' DO
       BEGIN

         FOR L := 1 TO 16 DO
         BEGIN
         WRITE (Data_file, T );
         WRITE (Data_file, Skip);
         WRITE (Data_file, A );
         WRITE (Data_file, Skip);
         WRITE (Data_file, L );
         WRITE (Data_file, Skip);
         WRITELN (Data_file, Billet_node  [ T, A , L ]:8:0);
         END; { L loop }

       END;  { A loop }

     END;  { T loop }

   CLOSE (Data_file);

END;   { Node dump }

{*****************************************************************************
*****************************************************************************}

{.pa}

PROCEDURE Arc_dump;

{ This procedure provides the displays and mechanices for
  dumping the transfer path percentagDe data to a disk file }

VAR

  Arc_type: STR_25;
  Ok_file: BOOLEAN;

BEGIN

  Clearscreen;
  Ok_file := false;

    IF NOT Replace THEN
    BEGIN

      REPEAT { until output filename is less than 8 letters)
      GotoRC (10,10);
      WRITELN ('Enter output filename (maximum of 8 letters)');
      Color ( red, white);
      GotoRC (13, 8);
      WRITELN (' UNLESS YOU DESIRE TO OVERWRITE PREVIOUS DATA, DO NOT USE ');
      Dirlist;
      Color ( white, blue);
      GotoRC (20, 13);
      READLN (Output_name);

        IF (Length (Output_name) > 8) THEN Wrong_answer

        ELSE
        BEGIN
```

```
        Ok_file := true;
        END;

      UNTIL Ok_file = true;

    Ok_file := false;
    Color ( white, blue);
    END;

  Blankline (24, 1);
  WRITE ('Dumping transfer path percentage data to the file ', Output_name);
  Skip := ' ';
  Output_name := CONCAT (Output_name,'.ard');
  Arcdata := Output_name;
  Strip (Arcdata);
  ASSIGN (Data_file,output_name);
  REWRITE (Data_file);

    FOR T := 1 TO 12 DO
    BEGIN

      FOR A := 'A' TO 'H' DO
      BEGIN

        FOR K := 'A' TO 'H' DO
        BEGIN
        WRITE (Data_file, T );
        WRITE (Data_file, Skip);
        WRITE (Data_file, A );
        WRITE (Data_file, Skip);
        WRITE (Data_file, K );
        WRITE (Data_file, Skip);
        WRITELN (Data_file,Transfer_path [ T, A , K ]:6:2);
        END;  ( K loop )

       END; ( A loop )

    END;  ( T loop )

  CLOSE (Data_file);

END;

{*************************************************************************
*************************************************************************}

{.pa}

PROCEDURE Length_dump;

{ This procedure provides the displays and mechanics for
  dumping the assignment tour length data from the program
  to a disk file }

VAR

   Ok_file: BOOLEAN;

BEGIN

  Clearscreen;
  Ok_file := false;

    IF NOT Replace THEN
    BEGIN

      REPEAT ( until output filename is less than 8 letters)
      GotoRC (10,10);
```

```
      WRITELN ('Enter output filename (maximum of 8 letters)');
      Color ( red, white);
      GotoRC (13, 8);
      WRITELN (' UNLESS YOU DESIRE TO OVERWRITE PREVIOUS DATA, DO NOT USE ');
      Dirlist;
      Color ( white, blue);
      GotoRC (20, 13);
      READLN (Output_name);

        IF (Length (Output_name) > 8) THEN Wrong_answer

        ELSE
        BEGIN
        Ok_file := true;
        END;

      UNTIL Ok_file = true;

    Ok_file := false;
    Color ( white, blue);
    END;

  Blankline (24, 1);
  WRITE ('Dumping billet lengths data to the file named : ', Output_name);
  T:= 1;
  Skip := ' ';
  Output_name := CONCAT (Output_name,'.led');
  Lengdata := Output_name;
  Strip (Lengdata);
  ASSIGN (Data_file, Output_name);
  REWRITE (Data_file);

    FOR T := 1 TO 12 DO
    BEGIN

      FOR A := 'A' TO 'H' DO
      BEGIN
      WRITE (Data_file, T );
      WRITE (Data_file, Skip);
      WRITE (Data_file, A );
      WRITE (Data_file, Skip);
      WRITELN (Data_file,Billet_length [ T, A ]:10);
      END;   ( A loop )

   END;  ( T loop )

 CLOSE (Data_file);

END; { length dump }

{*****************************************************************************
*****************************************************************************}

{.pa}

PROCEDURE Hills_dump;

{ This procedure provides the displays and mechanics for
  dumping the high limit data from the program to a disk file }

VAR

  Ok_file: BOOLEAN;

BEGIN

  Clearscreen;
  Ok_file := false;
```

```
    IF NOT Replace THEN
    BEGIN

      REPEAT ( until output filename is less than 8 letters)
      GotoRC (10,10);
      WRITELN ('Enter output filename (maximum of 8 letters)');
      Color ( red, white);
      GotoRC (13, 8);
      WRITELN (' UNLESS YOU DESIRE TO OVERWRITE PREVIOUS DATA, DO NOT USE ');
      Dirlist;
      Color ( white, blue);
      GotoRC (20, 13);
      READLN (Output_name);

        IF (Length (Output_name) > 8) THEN Wrong_answer

        ELSE
        BEGIN
        Ok_file := true;
        END;

      UNTIL Ok_file = true;

    Ok_file := false;
    Color ( white, blue);

    END; ( if not replace )

  Blankline (24, 1);
  WRITE ('Dumping high limit data to the file named : ', Output_name);
  Output_name := CONCAT (Output_name,'.hid');
  Hilidata := Output_name;
  Strip (Hilidata);
  ASSIGN (Data_file, Output_name);
  REWRITE (Data_file);

    FOR T := 1 TO 12 DO
    BEGIN

      FOR A := 'A' TO 'H' DO
      BEGIN
      WRITE (Data_file, T );
      WRITE (Data_file, Skip);
      WRITE (Data_file, A );
      WRITE (Data_file, Skip);
      WRITELN (Data_file, HiLimit [ T, A ]:10:0);
      END; ( A LOOP )

    END;  ( T LOOP )

  CLOSE (Data_file);

END;  ( Hilimit dump )

{*****************************************************************************
*****************************************************************************}

{.pa}

PROCEDURE Lolim_dump;

{ This procedure provides the display and mechanics for
  dumping the low limit data from the program to a disk file )

VAR

  Ok_file: BOOLEAN;
```

```
BEGIN

  Clearscreen;
  Ok_file := false;

    IF NOT Replace THEN
    BEGIN

      REPEAT ( until output filename is less than 8 letters)
      GotoRC (10,10);
      WRITELN ('Enter output filename (maximum of 8 letters)');
      Color ( red, white);
      GotoRC (13, 8);
      WRITELN (' UNLESS YOU DESIRE TO OVERWRITE PREVIOUS DATA, DO NOT USE ');
      Birlist;
      Color ( white, blue);
      GotoRC (20, 13);
      READLN (Output_name);

        IF (Length (Output_name) > 8) THEN Wrong_answer

        ELSE
        BEGIN
        Ok_file := true;
        END;

      UNTIL Ok_file = true;

    Ok_file := false;
    Color ( white, blue);
    END; ( if not replace )

  Blankline (24, 1);
  WRITE ('Dumping low limit data to the file named : ', Output_name);
  Output_name := CONCAT (Output_name,'.lod');
  Lolidata := Output_name;
  Strip (Lolidata);
  ASSIGN (Data_file, Output_name);
  REWRITE (Data_file);

    FOR T := 1 TO 12 DO
    BEGIN

      FOR A := 'A' TO 'H' DO
      BEGIN
      WRITE (Data_file, T );
      WRITE (Data_file, Skip);
      WRITE (Data_file, A );
      WRITE (Data_file, Skip);
      WRITELN (Data_file, Lolimit [ T, A ]:10:0);
      END; ( A LOOP )

    END;  ( T LOOP )

  CLOSE (Data_file);

END;  ( Lolimit dump )
```

APPENDIX M

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "ANSWERS.PAS"
{*************************************************************************
*************************************************************************}

{  This file contains the following procedures

    Invalid_answer
    Wrong_answer
    Correct_choice
    Correct_answer
    Another_change

}

PROCEDURE Invalid_answer;

{ This procedure provides the warning message on the bottom of the screen }

BEGIN

  Blankline (24, 1);
  Color ( white, red);
  WRITE (' Invalid answer, try again.');
  DELAY (2000);
  Color ( white, blue);
  Blankline (24, 1);

END; { Invalid_answer }

{*************************************************************************
*************************************************************************}


{.pa}

PROCEDURE Wrong_answer;

{ This procedure provides the warning message at the bottom of the screen }

BEGIN

  {make a red box}
  Color ( white, red);
  GotoRC (24, 15);
  WRITE (' Your response is incorrect, please try again.');
  Color ( white, blue);
  DELAY (2000);
  Blankline (24, 1);

END;  { Wrong_answer }

{*************************************************************************
*************************************************************************}


{.pa}


PROCEDURE Correct_answer;

{ The procedure evaluates the yes/no response }
```

```
BEGIN

  IF Answer in [ 'Y','y','N','n'] THEN  Ok_Answer := true
  ELSE
  BEGIN
  Ok_Answer := false;
  Wrong_answer;
  END;  { error condition if }

END; { Correct_answer}

{##########################################################################
##########################################################################}

{.pa}


PROCEDURE Correct_choice;

{ This procedure evaluates the response for correct choice }

BEGIN

  IF Choice in [ First .. Last] THEN Ok_choice := true
  ELSE
  BEGIN
  Ok_choice := False;
  Wrong_answer;
  END; { choice if}

END; { Correct_choice }


{##########################################################################
##########################################################################}


{.pa}

PROCEDURE Another_change;

{ This procedure prompts the user for another change }

BEGIN

  REPEAT  { Until answer correct }
  Blankline (24,1);
  WRITE (' Do you desire to change another? (Y/N) : ');
  READ ( Kbd, Answer);
  Ok_answer := true;
  Correct_answer ( Answer, Ok_answer);
  UNTIL Ok_answer = true;

END;    { Another_change }
```

APPENDIX N

```
THE FOLLOWING PROCEDURES ARE CONTAINED IN THE DISK FILE "STORDATA.PAS"
(*****************************************************************************
******************************************************************************)

{ This file contains the following procedures

     Save_data
     Replace_data          }

PROCEDURE Save_data;

{ This procedure provides the display and mechanics for
  storing data onto a disk file }

VAR

  Answer: CHAR;
  Ok_answer: Boolean;

BEGIN

  Finished := false;
  Center (21,'Do you desire to save any of the data to a disk file? (Y/N) ');
  READ (Kbd, Answer);
  Ok_answer := true;
  Correct_answer (Answer,Ok_answer);

    IF NOT Ok_answer THEN Save_data;

    IF Answer in [ 'Y','y' ] THEN
    BEGIN
    Clearscreen;
    GotoRC (10, 1);
    WRITELN;
    WRITELN ('   This procedure provides you with the opportunity to  ');
    WRITELN ('   save the data as presently configured in the program  ');
    WRITELN ('              to a disk file named by you.    ');
    Chg_var := 'Which data do you wish to save?';
    Choice_zero := 'None/finished saving data.';
    GotoRC (23,38);
    Pause;

      REPEAT {until finished }
      Clearscreen;
      Dchanges;

        IF Choice = '0' THEN Finished := true;
        IF Choice = '1' THEN Node_dump;
        IF Choice = '2' THEN Arc_dump;
        IF Choice = '3' THEN Length_dump;
        IF Choice = '4' THEN Hilim_dump;
        IF Choice = '5' THEN Lolim_dump;

      UNTIL Finished = true;

    Replace_data;

  END; { choice Y to save data }

END;  { save data procedure }
```

```
{##########################################################################
##########################################################################}

{.pa}

PROCEDURE Replace_data;

{ This procedure provides the displays and mechanics for
  replacing data in the default data files }

BEGIN

  Clearscreen;
  Center (10,'Now that you have saved that data to your own file, do you');
  Center (13,'desire to use any of it to replace the base data? (Y/N) ');
  READ (kbd, Answer);
  Clearscreen;
  Ok_answer := true;
  Correct_answer (Answer, Ok_answer);

    IF NOT Ok_answer THEN
    BEGIN
    Replace_data;
    END; { Wrong choice if }

    IF Answer in [ 'Y','y'] THEN
    BEGIN
    Clearscreen;
    Center (6,'   You have chosen to save the data as it is currently ');
    Center (7,'calculated by the program, to be the new initialization data.');
    Color ( red, white);
    Center (9,' This is a serious step.');
    Color ( white, blue);
    Center (11,' You will be losing the ');
    Center (12,' default data established in the model. ');
    GotoRC (23,30);
    Pause;

    REPEAT (until finished )
    Clearscreen;
    Chg_var := 'Which data file do you desire to change PERMANENTLY?';
    Choice_zero := 'None/finished replacing data.';
    Dchanges;

      IF Choice = '1' THEN Output_name := 'Nodes';
      IF Choice = '2' THEN Output_name := 'Arcs';
      IF Choice = '3' THEN Output_name := 'Length';
      IF Choice = '4' THEN Output_name := 'Hilimit';
      IF Choice = '5' THEN Output_name := 'Lolimit';

      IF Choice = '0' THEN Finished := true
      ELSE
      BEGIN

        REPEAT
        Clearscreen;
        Color ( red, white);
        GotoRC (12, 5);
        WRITE (' ARE YOU SURE YOU WANT TO REPLACE THE ', Output_name,' DATA? (Y/N) ');
        Color ( white, blue);
        READ (Kbd, Answer);
        Correct_answer (Answer, Ok_answer);
        UNTIL OK_answer = true;

        IF Answer in ['Y','y'] THEN
        BEGIN
        Replace := true;
```

```
          IF Choice = '1' THEN Node_dump;
          IF Choice = '2' THEN Arc_dump;
          IF Choice = '3' THEN Length_dump;
          IF Choice = '4' THEN Hilla_dump;
          IF Choice = '5' THEN Lolla_dump;

        Finished := false;
        Replace := false;

        END; { if answer is yes }

        IF Answer in ['N','n'] THEN Finished := false;

      END; { IF Choice = '0' THEN Finished := true;}

    UNTIL Finished = true;

  END; { choice = Y if }

END; { Replace data }
```

LIST OF REFERENCES

1. Unrestricted Line Officer Career Planning Guidebook (OPNAV 13-P-1), U.S. Government Printing Office, 1982

2. Howe, R. H., The Effect of PCS Policy Changes on Surface Warfare Officer Career Development, M. S. Thesis, Naval Postgraduate School, Monterey, CA, December 1984.

3. Zaks, Rodnay, Introduction to Pascal Including UCSD Pascal, Berkley: Sybex, 1981

4. Schneider, G. Michael; Weingert, Steven W.; and Perlman, David M., An Introduction to Programming and Problem Solving with Pascal, New York: John Wiley & Sons, 1981

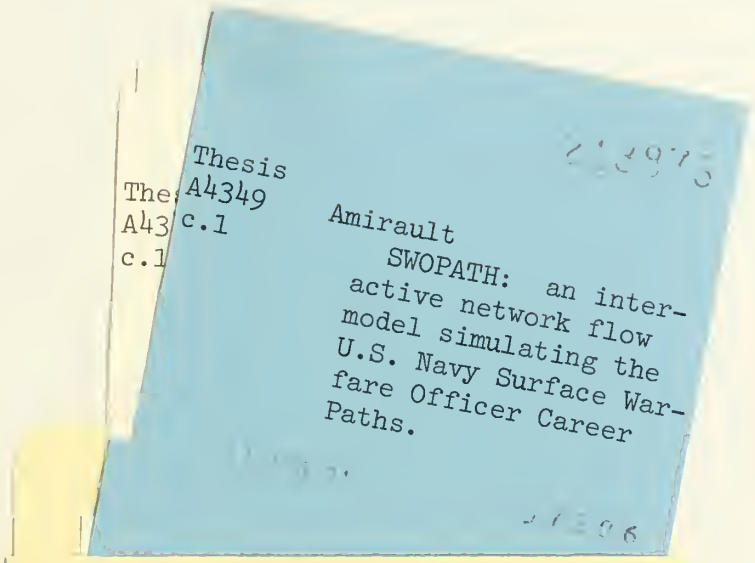
5. Schneider, G. Michael and Bruell, Steven C., Advanced Programming and Problem Solving With Pascal, New York: John Wiley & Sons, 1982

6. Turbo Pascal Version 3.0 Reference Manual, Scotts Valley: Borland International, 1985

7. Ferree, William Daniel, SWOTOURS: A Modification of an Interactive Computer Model to Analyze the Manpower Requirements of the Operational Tours of U.S. Navy Surface Warfare Officers, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1981

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Dr. Paul R. Milch, Code 55/Mh Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	5
4.	Deputy Assistant Secretary of the Navy for Manpower Office of the Secretary of the Navy Washington, D.C. 20350-2000	1
5.	CDR Don Nash (OP130E1) Office of DCNO (MPT) Department of the Navy Washington, D.C. 20370	1
6.	CDR Tom Halwachs, Code 30 Operations Analysis Programs Naval Postgraduate School Monterey, California 93943-5100	1
7.	LCDR Richard B. Amirault, USN 1205 Belvoir Lane Virginia Beach, Virginia 23464	3
8.	LCDR Wes Bergazzi (OP-130E4C) Office of DCNO (MPT) Department of the Navy Washington, D.C. 20370	2
9.	G. Thomas, Code 54 Naval Postgraduate School Monterey, California 93943-5100	1



Thesis
A4349
c.1
Amirault
SWOPATH: an inter-
active network flow
model simulating the
U.S. Navy Surface War-
fare Officer Career
Paths.